Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN ENXEÑARÍA DO SOFTWARE

Análise, deseño e implementación dunha Aplicación Web Java para a xestión de laboratorios de análises ambientais

Estudante: Anxo Varela Lema
Dirección: José Losada Pérez

A Coruña, xuño de 2020.

A miña nai, miña irmá e María

Agradecementos

Principalmente agradecerlle á miña familia e seres queridos polo apoio recibido e os ánimos dados que me axudaron a chegar ata aquí. Á súa vez, darlle as grazas ó director deste proxecto, José Losada Pérez, polos seus consellos e axuda.

Resumo

O obxectivo do proxecto é o deseño e implementación dunha aplicación web de xestión de laboratorios de análises ambientais. Dita aplicación permitirá aos usuarios xestionar todos os procedementos e información relativa a análises, mostras e resultados, ademais de levar a cabo unha xestión do tempo, aumentando así a eficiencia de ditas tarefas.

A aplicación resultante será unha plataforma web na que rexistrar ditos procedementos, resultando así nunha maneira automatizada e organizada de levar un control dos procesos.

A metodoloxía empregada neste proxecto é unha versión simplificada do *Proceso Unificado de Desenvolvemento Software*. Trátase dun desenvolvemento iterativo e incremental, no que en cada iteración se realiza un ciclo completo das diferentes fases de análise, deseño, implementación e probas. Desta forma, en cada iteración engádense funcionalidades ata acadar o obxectivo final.

En canto á fase de desenvolvemento, utilizaranse tecnoloxías enfocadas a unha arquitectura *Modelo-Vista-Controlador (MVC)*. Así, unha destas tecnoloxías será Java, xunto co framework Spring, máis concretamente Spring Boot. Mediante Spring Boot simplificarase o desenvolvemento do sistema que expón e implementa as operacións do servizo. Na parte da interface, utilizarase o framework de *Thymeleaf* xunto con *JavaScript*.

Abstract

The target of this project is the design and implementation of a web application for the management of environmental analysis laboratories. Said application will allow the users to manage all the procedures and relative information about analysis, samples, results, and keep track of time, thus increasing efficiency on these tasks.

The resulting application will be a web platform to register said procedures, thus resulting in automatic and organized way to keep control on these processes.

The methodology used in the project is a simplified version of the *Unified Software Development Process*. It is an iterative and incremental development, in which each iteration is performed a complete cycle of the different phases of analysis, design, implementation and test. By this, in each iteration new functionalities are added to achieve the final goal.

Regarding the development phase, technologies focused to an *Model-View-Controller (MVC)* architecture will be used. One of these technologies is Java, along with the Spring Framework, more concretely, Spring Boot. With Spring Boot, the development of the system, that exposes and implements the operation from the service, will be simplified. In the interface part, the *Thymeleaf* Framework will be used, along with *JavaScript*.

Palabras chave:

- Análise
- Tarefa
- Mostra
- Java
- Maven
- Spring
- Api
- Web
- Thymeleaf

Keywords:

- Analysis
- Task
- Sample
- Java
- Maven
- Spring
- Api
- Web
- Thymeleaf

Índice Xeral

Índice de Figuras

Índice de Táboas

Capítulo 1

Introdución

1.1 Motivación

Despois dun período realizando prácticas nun laboratorio de análise, puiden comprobar a problemática actual que padecen moitos laboratorios. Os procesos que se realizan nun laboratorio de análise son recorrentes e sinxelos de automatizar. Con todo, a maioría de laboratorios de pequeno ou mediano tamaño utilizan métodos cos cales perden eficiencia. Por exemplo, á hora de incubar unha mostra en estufa durante 48 horas, simplemente se engade unha etiqueta na mostra e é o encargado desta quen revisa todos os días que mostras se deben retirar. Outros problemas son manter unha lista de análises e probas que se realizan para cada cliente e a creación consecuente de informes entregables para dito cliente.

Dentro deste marco, este proxecto pretende crear unha aplicación web coa que os usuarios dos laboratorios poidan rexistrar as análises, mostras, etc., co fin de aumentar a organización eficiencia e, sobre todo, a facilidade coa que rexistrar e gardar e acceder aos datos. Do mesmo xeito, implementar unha funcionalidade de aviso automático da finalización da incubación dunha mostra ou do fin dunha tarefa. Por último, a posibilidade de exportar PDFs xerados a partires dos datos dunha análise.

1.2 Alcance e obxectivos

Este proxecto pretende crear un sistema que consiste nunha aplicación web, na que os usuarios dun laboratorio poidan rexistrar información relevante de análises, tarefas, mostras, etc. Á súa vez, preténdese tamén que ditos usuarios poidan xestionar o tempo de realización de ditas tarefas e as ferramentas necesarias para o envío de resultados para un cliente.

A continuación, expoñeranse as funcionalidades máis relevantes:

- **Xestión de usuarios.** Os usuarios poderán rexistrarse na aplicación, mediante dife-

rentes roles, podendo así acceder e modificar datos da aplicación, sempre que o seu rol llo permita.

- **Xestión de roles.** A aplicación dispoñerá de diferentes roles que proporcionarán diferentes funcionalidades a realizar. Diferénciase un usuario *"Lab manager"*, encargado do laboratorio e das funcionalidades máis do entorno de xestión, e un usuario *"employee"*, que se encargará de realizar as tarefas e procedementos pertinentes.
- **Xestión de análises.** Os usuarios poderán realizar e cumprimentar análises para diferentes clientes. As análises estarán formadas por diferentes tarefas.
- **Realización de tarefas.** Os usuarios realizarán tarefas, as cales son necesarias para completar as análises.
- **Seguimento de tarefas.** Mediante a integración da API de Google, os usuarios terán á súa disposición un control do tempo sobre as tarefas, desa forma, a través de correos electrónicos, avisarase a ditos usuarios de puntos chaves da tarefa, como por exemplo, en que momento se necesita retirar dunha estufa, cando se completa o decantado dunha mostra, etc.
- **Estado de tarefas e análises.** Mediante unha serie de etiquetas, saberase o estado dunha tarefa, por exemplo, "In progress". As análises necesitarán que todas as súas tarefas estean realizadas para proceder a cerrar dita análise.
- **Exportación de resultados.** Os resultados serán accesibles en todo momento para os usuarios. Estes poderán exportar os resultados en formato PDF, para o seu posterior envío ao cliente que os solicita.

1.3 Estrutura da memoria

A presente memoria estruturouse en 9 capítulos, 4 apéndices e un apartado de bibliografía. A continuación, explícase brevemente o seu contido:

- **Capítulo 1**: Capítulo no que se introduce o traballo, o seu alcance e motivacións.
- **Capítulo 2**: Capítulo no que se expoñen as principais ferramentas empregadas durante o desenvolvemento.
- **Capítulo 3**: Capítulo no que se analiza a viabilidade do proxecto.
- **Capítulo 4**: Capítulo no que se explica a arquitectura global do sistema e a metodoloxía empregada.

- **Capítulo 5**: Capítulo no que se realiza a planificación do proxecto estimando as tarefas.
- **Capítulo 6**: Capítulo no que se expoñen os requirimentos que debe cumprir o sistema.
- **Capítulo 7**: Capítulo no que se detalla o deseño realizado para a aplicación.
- **Capítulo 8**: Capítulo no que se explica a implementación e as probas coas ferramentas correspondentes.
- **Capítulo 9**: Capítulo no que se desenvolven as conclusións finais e posibles futuras liñas de traballo.
- **Anexo A**: API REST.
- **Anexo B**: Manual de usuario.
- **Relación de acrónimos**.
- **Glosario**.
- **Bibliografía**.

Capítulo 2

Ferramentas e tecnoloxías utilizadas

Neste capítulo explicaranse de maneira resumida as tecnoloxías e ferramentas utilizadas para a realización do proxecto.

2.1 Linguaxes de programación

2.1.1 Java

Java [1] é unha linguaxe de programación de propósito xeral, concorrente, baseada en clases e orientada a obxectos. É unha linguaxe de estrutura sinxela, polo que pode ser utilizada polo maior número de usuarios de forma fluída. Pode ser executada en diversos sistemas operativos e está pensada principalmente para entornas de produción.

A linguaxe é de alto nivel. Inclúe xestión automática de almacenamento, con un colleitador de lixo para evitar así problemas de seguridade da desasignación explícita. *Java* non inclúe estruturas inseguras, como pode ser o acceso a *arrays* sen comprobar índices.

Java Platform é unha plataforma de desenvolvemento de aplicacións Java para o ámbito empresarial. O obxectivo desta ferramenta é prover ós desenvolvedores dunha potente colección de *APIs* para acortar o tempo de desenvolvemento e reducir a complexidade da aplicación.

2.1.2 JavaScript

JavaScript (JS) [2] é unha linguaxe lixeira e interpretada, orientada a obxectos con funcións de primeira clase, máis coñecido como a linguaxe de script para páxinas web, pero tamén usado en moitas entornas sen navegador. Isto quere dicir que as páxinas web xa non teñen que ter só *HTML* estático, senón que tamén poden incluír programas que interactúen co usuario e controlen ó navegador creando contido de forma dinámica. É unha linguaxe script multi-paradigma, baseada en prototipos, dinámica, soporta estilos de programación funcional,

orientada a obxectos e imperativa.

A definición estándar da linguaxe *JavaScript* ven definida por ECMA, en concreto na súa especificación ECMA-262. Mentres que a especificación do *DOM* foi publicada pola *World Wide Web Consortium (W3C)*, que estandariza as características que debe soportar un navegador web en relación co seu *DOM*.

2.1.3 HTML

HTML (HyperText Markup Language) [3] é unha linguaxe de marcado de hipertexto. É o estándar na visualización de páxinas web e o que todos os navegadores actuais adoptaron. Orixinalmente, *HTML* foi deseñado como unha linguaxe para describir semanticamente documentos científicos. Con todo, o seu deseño xeral permitiu que se adapte, nos anos posteriores, para describir unha serie de outros tipos de documentos e incluso aplicacións.

É un estándar a cargo do *World Wide Web Consortium (W3C)*. *HTML* considerase a linguaxe web máis importante. É unha linguaxe lixeira que se estrutura en forma de etiquetas.

2.1.4 CSS

CSS (Cascading Style Sheets) [4] é a linguaxe utilizada para describir a presentación de documentos HTML ou XML, isto inclúe varias linguaxes baseadas en XML, como son XHTML ou SVG. Describe como debe ser renderizado o elemento estruturado en pantalla, en papel ou en outros medios.

CSS pertence a base de linguaxes da Open Web e posúe unha especificación estandarizada por parte do *W3C*.

2.1.5 LaTex

Latex [5] é un sistema de preparación de documentos para a composición tipográfica de alta calidade. Normalmente emprégase para documentos técnicos ou científicos de tamaño mediano ou grande, pero é posible empregalo para calquera forma de publicación.

Latex está baseado na filosofía de animar aos autores, de forma que estes se centren máis no contido do documento que na aparencia e estilos.

2.2 Frameworks e librerías

2.2.1 Spring

Spring Framework [6] é unha solución lixeira destinada a construción de aplicacións Java centrándose só na propia aplicación, delegando en *Spring* o manexo da infraestrutura. Con todo, Spring é modular, permite usar só as partes que se precisan, sen ter que depender do resto.

Permite empregar a Inversión de Control (IoC) e soporta a xestión de forma declarativa da transicionalidade, o acceso remoto á lóxica a través de RMI e servizos web e diferentes opcións para preservar os datos. Ofrece un marco MVC completo e permite integrar AOP (Aspect Oriented Programming) de forma transparente no software.

Spring está composto por diferentes características organizadas en 20 módulos. Estes agrúpanse en 6 principais: Core Container, Data Access/Integration, Web, AOP, Instrumentation e Test.

Neste proxecto empregáronse os seguintes módulos:

- **Core.** Están agrupados no *Core Container*. Proporcionan as partes fundamentais, como son a IoC (*Inversion of Control*) ou a inxección de dependencias.
- **Web.** Encóntrase na parte web de *Spring*. É un módulo dentro do proxecto que proporciona a integración básica de características relacionadas coa web, como poden ser a inicialización do contedor da inversión de control mediante *servlets*.
- **Test.** Centrado na creación e execución de tests unitarios de *JUnit*. Grazas a este módulo, dispoñemos de entorna de probas, con posibilidade de datos *Mock*, sen necesitar servidores de Produción.

Spring Boot

Spring Framework co paso dos anos comezou a volverse demasiado complexo. É preciso pasar por un largo proceso de requisitos para comezar un novo proxecto.

Un dos principais problemas é o proceso de configuración. Inicialmente empregábanse *XMLs*, aínda que en futuras versión se engadeu a configuración baseada en *Java*, como unha opción máis sinxela aos *XMLs*. Algúns destes procesos de configuración son o manexo da transaccionalidade, Spring MVC, *servlets*, etc.

A pesar das continuas melloras, estes procesos de configuración eran un proceso necesario para o comezo do proxecto e durante o mesmo. Isto coñécese como ”fricción do desen-

volvemento", é dicir, todo o tempo que se dedica á configuración que non pertence á lóxica de negocio, ó desenvolvemento propiamente dito da aplicación. *Spring Boot* [7] cambia este proceso.

Spring Boot facilita a creación de aplicacións *stand-alone* baseadas en *Spring*. Encárgase do proceso de configuración, de xeito que o usuario poida centrarse na lóxica de negocio. A pesar disto, sempre se necesita un pequeno grao de configuración.

Algunhas das vantaxes son:

- Reduce o tempo de desenvolvemento, aumentando desa forma a produtividade.
- Evita especificar anotacións e configuracións XML.
- Sinxeleza á hora de integrarse con ecosistemas de Spring, *ORMs*, *Spring Security*, etc.
- Proporciona servidores HTTP, como Tomcat ou Jetty.
- Proporciona CLI (Command Line Interface) para desenvolver ou probar as aplicacións dende un prompt.
- Inclúe *plugins* para traballar con bases de datos embebidas ou en memoria.

2.2.2 Thymeleaf

Thymeleaf [8] é un moderno motor de modelos *Java Server-Side*, tanto para web como para entornas *Standalone*, capaz de procesar *HTML, XML, JavaScript, CSS* ou texto plano.

O obxectivo principal de *Thymeleaf* é proporcionar unha forma elegante e moi mantible de crear modelos. Para conseguir isto, baséase no concepto de *Plantillas Naturais* para inxectar a súa lóxica en ficheiros dunha forma que non afecta ó modelo de ser empregado como un prototipo de deseño. Grazas a isto, mellórase a comunicación do deseño e redúcese o espazo entre os equipos de desenvolvemento e de deseño.

2.3 Ferramentas de desenvolvemento

2.3.1 Eclipse IDE

Eclipse [9] é unha plataforma software composta por un conxunto de ferramentas de código aberto para crear entornas de desenvolvemento integrado (IDE), que se poden empregar para crear aplicacións como sitios web, programas JAVA, C++, etc.

Creada inicialmente por IBM, pero actualmente mantida pola Fundación Eclipse, unha organización independente sen ánimo de lucro.

Unha das características principais do IDE Eclipse é a de poder ampliar as súas capacidades mediante módulos (*plugins*), en contra doutro tipos de entornas monolíticas.

O presente proxecto está construído totalmente en Eclipse IDE.

2.3.2 MAVEN

Apache Maven [10] é unha ferramenta para automatizar a construción, principalmente de obxectos Java. *Maven* aborda dous aspectos do software de construción. En primeiro lugar, describe como se constrúe un software e, en segundo lugar, describe as súas dependencias. Emprega convencións para o procedemento de compilación.

Un ficheiro XML (POM) describe o proxecto de software que se está a construír, as súas dependencias doutros módulos e compoñentes externos, a orde de compilación, directorios e complementos necesarios. Inclúe obxectivos predefinidos para realizar certas tarefas ben definidas, como a compilación de códigos e o seu embalaxe.

Unha das características principais é a descarga de forma dinámica das bibliotecas Java e dos complementos Maven dun ou varios repositorios, como o *Maven Central Repository*, e o gardado localmente.

2.3.3 GIT

Git [11] é un sistema de control de versións distribuído, caracterizado principalmente pola súa velocidade, rendemento, flexibilidade e usabilidade. Deseñado por Linus Torvalds, foi creado inicialmente co propósito de ser o sistema de control de versións para o Kernel de Linux.

2.3.4 MySQL Workbench

MySQL Workbench [12] é unha ferramenta visual unificada para arquitectos, desenvolvedores e administradores de bases de datos. Prové un modelado de datos, desenvolvemento SQL e ferramentas sinxelas de administración, entre outras, a configuración de servidores, administración de usuarios, backups, etc.

2.3.5 Overleaf

Overleaf [13] é unha empresa social que constrúe modernas ferramentas de creación colaborativa para científicos. O produto principal é un editor colaborativo en liña en tempo real

para traballos, teses, informes técnicos e outros documentos escritos na linguaxe de marcaxe LaTeX.

Overleaf viu unha rápida adopción en ciencia e investigación, utilizada por importantes institucións como Stanford e Caltech. Ademais da utilización por parte de investigadores, tamén pasou a formar parte de moitas institucións académicas.

O obxectivo principal é abordar os problemas que xorden a hora de redactar artigos de colaboración e poder facer máis accesible LaTeX para usuarios de calquera nivel.

2.4 Sitemas de Xestión de Bases de Datos

2.4.1 MySQL

MySQL [14] ofrece un servidor de base de datos SQL *(Structured Query Language)* moi rápido, multithreaded, multi-usuario e robusto. Está deseñado para os sistemas de produción de gran carga e críticos, así como para incorporar a software despregado de forma masiva. É posible empregar MySQL baixo dúas licenzas distintas, pode ser empregado como un produto Open Source, baixo as condicións da licenza GNU *(General Public License)*, ou pode ser adquirida a versión comercial estándar de Oracle.

2.5 APIs

2.5.1 Google Calendar API

A *API de Calendario de Google* [15] permite mostrar, crear e modificar eventos do calendario de Google, así como traballar con moitos outros obxectos relacionados co calendario, como calendarios ou controis de acceso.

É unha *API REST* á que se pode acceder a través de chamadas *HTTP* explícitas ou a través das bibliotecas de clientes de Google. Mediante esta API, podemos acceder á maioría de funcionalidades dispoñibles na interface web de Google Calendar.

Os principais recursos aos que se poden acceder son:

- **Eventos**. Un evento no calendario contén información como o título, datas de inicio e fin ou os participantes de dito evento.
- **Calendarios**. Un calendario é unha colección de eventos.
- **Lista de calendarios**. Contén nunha lista todos os calendarios do usuario.

2.6 Outros

2.6.1 OAuth 2.0

OAuth 2.0 [16] é un protocolo estándar para a autorización. Céntrase na sinxeleza do desenvolvedor do cliente ao tempo que proporciona fluxos de autorización específicos para aplicacións web, aplicacións de escritorio, teléfonos móbiles, etc. Esta especificación e as súas extensións están a desenvolverse dentro do *ETF OAuth Working Group*.

2.6.2 HTTP

O protocolo de transferencia de hipertexto (*HTTP*) [17] é un protocolo a nivel de aplicación para sistemas de información hipermedia distribuídos e colaborativos. É un protocolo xenérico, sen estado, que se pode usar para moitas tarefas máis aló do seu uso para hipertexto, como servidores de nomes e sistemas de xestión de obxectos distribuídos, a través da extensión dos seus métodos de solicitude, códigos de erro e cabeceiras.

HTTP emprégase na iniciativa de información global World-Wide Web dende 1990. Esta especificación define o protocolo denominado ”HTTP / 1.1”.

2.6.3 REST

Representa l State Transfer (*REST*) [18] é un estilo de arquitectura Web con unhas características e restricións definidas por Roy Fielding na súa tese doutoral.

Na actualidade, é moi sinxelo atopar arquitecturas REST APIs de servizos web modernos. Unha API REST consiste nun conxunto de recursos interconectados, sobre os cales se poden realizar diferentes operacións estándar.

Capítulo 3

Análise de viabilidade

Neste capítulo expoñerase a análise de viabilidade do proxecto levada a cabo.

A análise de viabilidade é un proceso fundamental para calquera tipo de proxecto, xa que serve para determinar as probabilidades de finalizar este con éxito.

Lévase a cabo antes do comezo da realización do proxecto en si, debido a que dependendo de dita análise, valórase, de forma real, a viabilidade do mesmo. É un proceso crítico.

Dita análise está dividida en tres puntos: **económica**, **técnica** e **de mercado**.

3.1 Viabilidade económica

En primeiro lugar, realízase a viabilidade económica do proxecto. Existen tres costes principais nos que apoiar dita viabilidade, estes son:

- Recursos humanos.
- Recursos software.
- Recursos hardware.

Os **recursos humanos** no caso deste proxecto redúcense a unha persoa que se encarga de todas as fases do desenvolvemento, polo que o coste é baixo.

En relación aos **recursos software**, como o obxectivo deste proxecto é a realización dunha aplicación web, existen múltiples opcións de software *open source* adecuado para dita realización. Empregando este tipo de software libre, conseguimos que os costes se reduzan substancialmente, de maneira moi sinxela.

Por último, en canto aos **recursos hardware**, tan só será necesario un ordenador con conexión a internet.

Unha vez estudado o impacto dos tres anteriores custos, podemos asumir que o proxecto é viable economicamente, debido a unha inversión necesaria baixa.

3.2 Viabilidade técnica

Como se viu co capítulo 2, as ferramentas e tecnoloxías empregadas na realización do proxecto son amplamente coñecidas e utilizadas na comunidade de desenvolvemento de software, polo que supón un risco moi baixo. Outro punto a ter en conta é a previa experiencia de ditas tecnoloxías por parte do desenvolvedor, facendo así, se cabe, máis reducido o risco.

Tendo en conta isto, podemos asumir que, tecnicamente, o proxecto é viable.

3.3 Viabilidade de mercado

Na actualidade, existen diversas opcións de *software* de xestión de laboratorios, como poden ser *Lockbox LIMS*, *Quartzy* ou *LabCollector*.

A maioría destes produtos céntranse na parte administrativa dos distintos procesos. *Lockbox LIMS* está máis orientado á trazabilidade de mostras, á xestión do almacenamento de ditas mostras ou ós diferentes protocolos que se realizan. Por outra parte, a funcionalidade de *Quartzy* reside maioritariamente na xestión de compras dentro do laboratorio de produtos necesarios para a realización das probas sobre as mostras. Por último, *LabCollector* é similar a *Lockbox LIMS* en canto a trazabilidade ou almacenamento das mostras, pero máis centrada en análises biolóxicas. A pesar disto, presenta maior xestión en canto ás actividades realizadas polos usuarios.

Algunhas características presentes neste tipo de aplicacións son o inventariado dos recursos do laboratorio ou os datos de calibración de aparatos.

A aplicación desde proxecto céntrase máis nas actividades ”diarias” que realizan os usuarios dun laboratorio. A forma na que se xestionan estas actividades baséase en aplicacións de xestión de software, como poden ser *JIRA de Atlassian* ou *Redmine*, conseguindo así unha xestión dun laboratorio centrada en análises e tarefas, é dicir, unha plataforma centralizada na que un usuario pode realizar un seguimento do seu traballo, comprobar resultados ou exportar informes.

Debido á usabilidade da aplicación e ó enfoque centrado no seguimento e xestión das tarefas dos usuarios, pódese afirmar que o desenvolvemento do proxecto é viable no mercado, principalmente orientado á pequena e mediana empresa.

Capítulo 4

Introdución ao desenvolvemento realizado

Neste capítulo explicarase tanto a arquitectura global do sistema, como a metodoloxía empregada.

4.1 Arquitectura global do sistema

A arquitectura do sistema creouse empregando o patrón de deseño **Modelo-Vista-Controlador** ou *MVC* [19, 20]. Esta arquitectura ten unha ampla aceptación para o desenvolvemento de software corporativo. Pretende dividir o sistema en tres capas diferentes que se encargan da lóxica de control da interface e o acceso a datos. Isto facilita o mantemento e a evolución dos sistemas segundo a independencia das clases actuais en cada capa. Emprégase amplamente en aplicacións web nas que un usuario iteractúa co sistema e este responde.

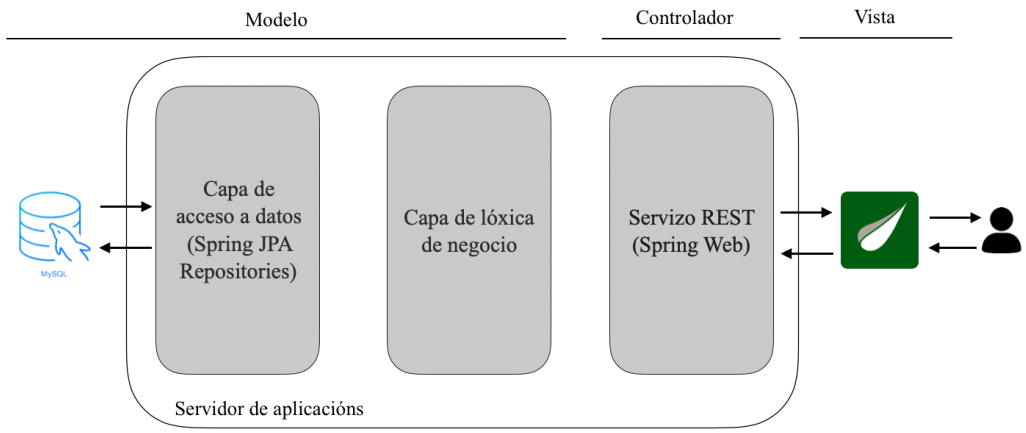


Figura 4.1: Esquema MVC

Como o seu propio nome indica, conta con tres módulos diferentes. Por unha parte, atopamos o módulo **Vista**, que se encarga de presentarlle ao usuario a información do sistema e responder as súas solicitudes. Por outra parte, encontramos o **Controlador**, cuxa principal función é a de procesar as peticións realizados polo usuario ao interactuar co sistema. O Controlador encárgase da comunicación entre as capas da Vista e do Modelo. Por último, está o **Modelo**, que é a parte do sistema que contén tanto a información que mostra a Vista como a lóxica de negocio.

Como se aprecia na figura 4.1, dentro das partes do patrón MVC, dividiuse o sistema en diferentes grupos lóxicos. Dentro destes grupos, empréganse distintas tecnoloxías, linguaxes e frameworks. A continuación, explicaranse cada un dos grupos:

- **Capa de acceso a datos.** É a capa lóxica do subsistema que se encarga de acceder a base de datos, *MySQL* no caso deste proxecto, e recupera todos os datos necesarios para a seguinte capa, a capa de lóxica de negocio. Esta capa está codificada en Java e emprégase a tecnoloxía Spring JPA (*Java Persistance API*), coa que podemos *mapear* os obxectos Java ás entidades de base de datos e realizar operacións sobre eles a través dos repositorios.

- **Capa de lóxica de negocio.** Nesta capa desenvólvese a lóxica interna de casos de uso e funcionalidades da aplicación. Está desenvolta empregando Java como linguaxe de programación, apoiándose en Spring para o manexo de dependencias e transaccionalidade. Emprega os repositorios que se comunican coa anterior capa de acceso a datos.

- **Servizo REST.** Encárgase de realizar as comunicacións entre a vista e a capa de lóxica de negocio. Recibe peticións HTTP provenientes da vista, procésaa axudándose da capa de lóxica de negocio, que a súa vez se axuda da capa de acceso a datos, e devólvelle o resultado de novo á vista. Está desenvolta en Java con axuda do módulo Spring Web MVC de Spring, encargado da anotación dos métodos dos controladores que actuarán como manexadores das distintas operación definidas. O formato de ditas respostas é en *JSON*.

- **Vista.** Na capa da vista atópase a lóxica da interface de usuario. Encárgase de procesar as peticións que o usuario realiza sobre a interface, delegando no servizo REST para isto, e unha vez recibido a resposta do servizo REST, devólvella ao usuario. Esta capa está desenvolta utilizando *HTML* como linguaxe, *CSS* para os formatos, *JavaScript* para funcións dentro da páxina e co Framework *Thymeleaf* para o acceso a datos.

4.2 Metodoloxía empregada

A metodoloxía empregada neste proxecto software é unha versión simplificada, e máis específica, do Proceso Unificado de Desenvolvemento Software, denominada *Rational Unified Process (RUP)* [21].

4.2.1 Rational Unified Process

O Proceso Unificado do Software *(RUP)* é un framework para procesos de enxeñaría software desenvolto polo *Rational Software*. Está composto pola acumulación de coñecementos e boas prácticas aportadas por numerosos colaboradores ao largo de moitos anos, nunha ampla variedade de situacións. Ao aplicar este proceso pódese producir software de alta calidade que satisfaga as necesidades dos usuarios finais dentro dun presuposto e datas predicibles.

RUP emprega unha aproximación **iterativa**, é dicir, unha secuencia de pasos incrementais ou iteracións. Cada iteración inclúe algunha, ou a maioría, das disciplinas de software (requisitos, análise, deseño, implementación, etc.). Cada iteración ten ben definidos un conxunto de obxectivos e produce unha parte parcial do traballo a implementar no sistema final. A seguintes iteracións baséanse nas iteración anteriores e constrúense en base a elas, para poder así refinar o sistema.

Outro aspecto importante é o de definir as bases da arquitectura do sistema software que se var crear, debido a que esta acción permite minimizar os riscos inherentes a ela.

Fases do Proceso Unificado de Software

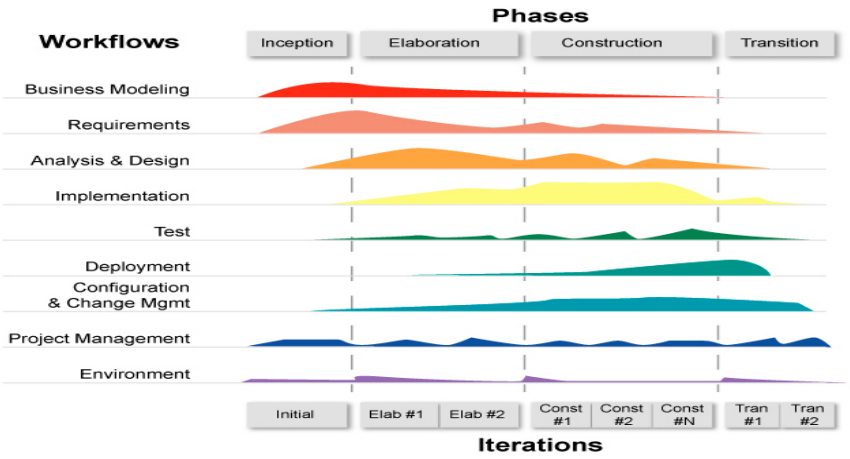


Figura 4.2: Fases do Proceso Unificado do Software

Como podemos observar na figura 4.2, o Proceso Unificado de Software divídese en dúas dimensións:

- A dimensión horizontal representa a parte dinámica ou dimensión temporal do proceso. Expresa ciclos, fases e iteracións.
- A dimensión vertical representa a estrutura estática do proceso. Describe como os elementos do proceso (actividades, disciplinas, artefactos e roles) son agrupados dunha forma lóxica nos fluxos de traballo.

Dentro da estrutura horizontal, é dicir, a dinámica, encontramos o ciclo de vida dun proxecto. O RUP prové unha aproximación ao desenvolvemento iterativo dividindo o proxecto en catro fases:

- **Comezo.** Establece unhas pautas de que sistema construír, apoiándose en requisitos de alto nivel e establecendo un alcance do sistema. Minimiza un gran número de riscos de negocio, poñéndose todas as partes de acordo para decidir se o proxecto se leva a cabo ou non.
- **Elaboración.** Nesta fase lévanse a cabo as tarefas técnicas, como o deseño, implementación ou tests, e establécese unha arquitectura executable e estable tendo en conta o alcance do proxecto. Minimízanse os principais riscos técnicos, implementando e validando o código actual.
- **Construción.** Realízase a maior parte da implementación, como resultado de moverse dende unha arquitectura executable a unha primeira versión operativa do sistema. Despréganse varias versións alpha ou internas para asegurarse de que o sistema cumpre coas necesidades dos usuarios. A fase termínase lanzando unha versión beta completamente funcional do sistema, incluíndo a documentación de instalación e soporte.
- **Transición.** Asegúrase que o software cumpre coas necesidades dos usuarios. Realízanse tests do produto de cara ao lanzamento e para facer axustes menores baseados no *feedback* dos usuarios, como poden ser a configuración, instalación ou problemas de uso.

4.2.2 Iteracións

Mediante a metodoloxía descrita no punto anterior, dividiuse o proxecto en 12 iteracións.

En cada iteración realizáronse as fases de análise, deseño, implementación e probas. Cada unha de elas incorpora novas funcionalidades á anterior. A continuación, explicaranse brevemente.

1. **Iteración 1: Definición do proxecto**
   Nesta primeira fase lévase a cabo a análise de viabilidade e a definición do dominio do proxecto. Ao mesmo tempo, tamén se realiza o estudo sobre as tecnoloxías dispoñibles que se empregarán para a construción do mesmo, de xeito que se minimicen os riscos inherentes do descoñecemento de ditas tecnoloxías.

   Por último, redáctanse de maneira pouco formal os requisitos e a arquitectura global do sistema.

2. **Iteración 2: Posta en marcha do proxecto**
   Tras a realización e definición xeral do proxecto, formalizáronse, refináronse e detalláronse por escrito os casos de uso que conterá o proxecto.

   Ademais do anterior, outra tarefa pertencente a esta iteración é a da construción da entorna de traballo no que se desenvolverá a aplicación. Ao finalizar, terase unha entorna de traballo funcional, con un arquetipo coas dependencias básicas necesarias e as tecnoloxías e ferramentas que se precisen de cara ao futuro desenvolvemento.

3. **Iteración 3: Xestión de usuarios**
   Nesta iteración desenvolveuse o subsistema que xestiona aos usuarios. En primeiro lugar, creáronse as entidades persistentes e as operacións necesarias sobre eles, tanto en repositorios como servizos. As fases que se levan a cabo son de análise, deseño, implementación e probas sobre esta primeira versión, que permite o rexistro e aceso de usuarios.

   As características principais son:

   - Rexistro de usuarios, seleccionando o rol que se precise e proporcionando os datos persoais necesarios.
   - Acceso a usuarios existentes á aplicación.
   - Modificación dos datos persoais dos usuarios.

4. **Iteración 4: Xestión de laboratorios**
   Desenvólvese o subsistema que xestiona os laboratorios. Como na iteración anterior, créanse as entidades persistentes necesarias e as operación sobre estas. Partindo da iteración anterior, mediante o rol asignado a cada usuario, pódese crear ou buscar laboratorios.

   As principais funcións son:

   - Creación de laboratorios por parte dos usuarios que teñan o rol de *'LAB_MANAGER'*.

- Procura de laboratorios por parte dos usuarios con rol *'EMPLOYEE'*.

5. **Iteración 5: Xestión de notificacións**
   Nesta quinta iteración, desenvólvese a xestión de notificacións. Mediante estas notificacións, un usuario *'EMPLOYEE'* pode solicitar a asignación a un laboratorio e un usuario *'LAB_MANAGER'* pode asignar a dito usuario ao laboratorio.

6. **Iteración 6: Xestión de análises**
   Nesta iteración, procédese a desenvolver a xestión de análises. En primeiro lugar, créanse as entidades persistentes necesarias e as operacións sobre estas. Esta xestión consiste na posibilidade de crear análises cos datos pertinentes. Ademais da creación, tamén se permite buscar análises específicas, filtrar dita procura polo estado da análise (*'OPEN'* ou *'CLOSE'*) e acceder a unha análise para obter máis detalles sobre ela.

   Como nas anteriores iteracións, realízanse as catro fases do proceso de desenvolvemento para levala a cabo.

7. **Iteración 7: Xestión de tarefas**
   Baseándose na iteración anterior, nesta iteración levarase a cabo a xestión de tarefas. En primeiro lugar, créanse as entidades persistentes necesarias e as operación sobre estas.

   Unha vez dentro dunha análise, permíteselle ao usuario crear tarefas relacionadas con dita análise. Despois de crear a tarefa, existe a posibilidade de acceder a dita tarefa para máis detalle.

   Dentro do detalle da tarefa, as funcionalidades activas desta iteración son a de mostrar a información descritiva da tarefa (nome, data de creación, estado, importancia, etc.), así como as descrición da mesma.

8. **Iteración 8: Xestión de comentarios**
   Nesta iteración de xestión de comentarios, lévase a cabo a funcionalidade de comentar dentro dunha tarefa.

   Dentro das posibles accións atopamos:

   - Creación de comentarios.
   - Eliminación de comentarios. Para poder eliminar un comentario, é condición obligatoria que sexa o mesmo usuario que creou o comentario.

9. **Iteración 9: Xestión de mostras**
   O seguinte paso ven dado pola iteración de xestión de mostras. En primeiro lugar, créanse as entidades persistentes do sistema e as operacións necesarias sobre elas.

Procédese a creación de mostras, cun nome e unha cantidade. Ditas mostras visualízanse na vista das tarefas, coa posibilidade de engadir cantas mostras como sexan necesarias.

10. **Iteración 10: Integración con Google (Login e API Calendar)**
    Esta décima iteración corresponde coa integración con Google que se divide en dúas partes:

    - En primeiro lugar, lévase a cabo a posibilidade de *logearse* mediante unha conta de Google.

      Solicitouse un cambio na forma de acceder os usuarios á aplicación. Ata o momento, o procedemento consistía en inserir un correo electrónico e un contrasinal, en cambio, a partir desta iteración o procedemento realizarase a través dunha plataforma externa, neste caso Google.
    - En segundo lugar, engádese a funcionalidade de anotar eventos a través da *API Calendar* de Google.

      Procédese a levar a cabo unha funcionalidade, mediante a cal os usuarios poden seleccionar unha data de finalización, cun texto descritivo, para unha tarefa. Unha vez seleccionada dita data, o sistema comunícase co calendario da conta de Google do usuario, mediante a *API Calendar*, anotando así un evento correspondente a dita data.

      Os datos que se anotan no evento son o identificador da tarefa, para que o usuario poida identificala con facilidade, e o texto descritivo introducido polo usuario, de forma que resuma a actividade a realizar coa tarefa.

11. **Iteración 11: Exportación de resultados en PDFs**
    Nesta iteración lévase a cabo a funcionalidade de engadir resultados ás mostras e ás análises. Para isto créanse as entidades persistentes necesarias.

    No caso das mostras, o resultado será numérico, en cambio, as análises teñen tres posibles resultados, *PASS*, *DETECTION* e *FAIL*.

    Unha vez que as mostras e análises teñen os seus respectivos resultados, engádese a funcionalidade de exportar unha análise a PDF. Este PDF conterá a a seguinte información:

    - Información da análise como o identificador, nome, resultado, responsable, cliente, etc.
    - Información das tarefas que se realizaron sobre a análise.
    - As mostras que contén cada tarefa e o resultado numérico.

12. **Iteración 12: Elaboración da memoria**
    Nesta última iteración final, procédese a realización da memoria do proxecto empregando a ferramenta LaTeX, debido a súa potencia e eficacia de cara a proxectos académicos.

Capítulo 5

Planificación e análise de custos

Os dous elementos máis importantes de cara á realización dun proxecto de desenvolvemento software son o **tempo** e o **custo**. O labor do *Xefe do Proxecto* é a de coñecer ditos parámetros en profundidade de maneira previa ao inicio do desenvolvemento. Para coñecer estes parámetros, é necesario realizar un estudo de planificación e análises de custos. Mediante este estudo, é posible minimizar ambos parámetros, conseguindo así o resultado máis óptimo.

5.1 Recursos

Centrámonos en dous tipos de recursos principais: humanos e técnicos.

5.1.1 Recursos humanos

De cara a realización do proxecto, simúlanse tres tipos de perfís: Xefe de proxecto, Analista e Programador. Non obstante, detrás destes perfís está o propio autor do proxecto en colaboración co titor asignado.

Detallamos os perfís a continuación:

- **Xefe de Proxecto.** É o encargado da xestión e coordinación do proxecto. Esta xestión inclúe ós recursos. Mediante a súa participación conséguese un desenvolvemento óptimo no proxecto. Outra das súas principais funcións é a de definir os requisitos do sistema, realizar a planificación e o seguimento.

- **Analista.** O analista é o encargado de tomar os requisitos definidos polo xefe do proxecto e realizar unha análise técnica e funcional. Nas fases de desenvolvemento o analista ten a función de comprobar que todos os requisitos necesarios da iteración se cumpran correctamente, así como proporcionar guías e apoio aos programadores.

- **Programador.** É o rol que se encarga, mediante os documentos funcionais xerados polo analista, de desenvolver a aplicación. Outra función do programador é a de realizar as probas pertinentes.

5.1.2 Recursos técnicos

Dentro dos recursos técnicos do desenvolvemento do proxecto, engádese un ordenador portátil coas seguintes especificacións:

- **Marca e modelo:** Apple MacBook Pro 2018.
- **Procesador:** 2,3 GHz Intel Core i5.
- **Memoria RAM:** 8 GB 2133 MHz LPDDR3.

No ordenador instálase o software e ferramentas necesarias para levar a cabo o desenvolvemento do proxecto.

5.2 Planificación e custos

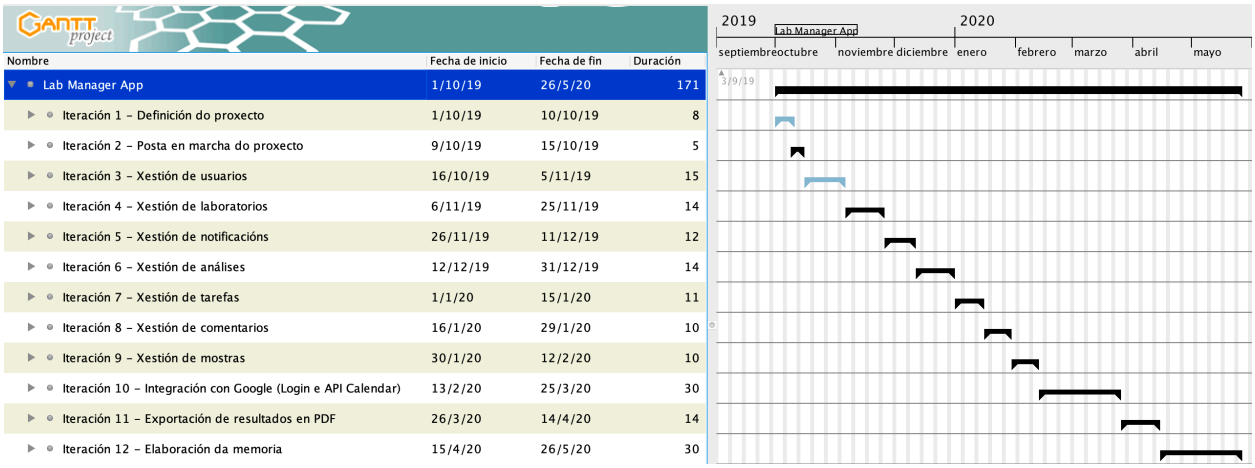
Mediante as iteracións definidas anteriormente, no apartado 4.2.2, pódese levar a cabo a planificación do proxecto. Estas iteracións dividíronse en tarefas de menor tamaño e definíronse para estimar o gasto en tempo e custo que implica o desenvolvemento.

En primeiro lugar, para poder realizar a determinación do custo das tarefas de cada iteración, é necesario asignar un prezo/hora a cada recurso:

Recurso	**Custo**
Xefe do proxecto	33€/h
Analista	24€/h
Programador	16€/h
Ordenador	2.000€

A partir deste punto, no que se coñecen os custos de cada recurso, procédese a realizar un diagrama de Gantt, coa finalidade de ilustrar de forma gráfica a duración do proxecto, coñecendo así o tempo empregado no mesmo.

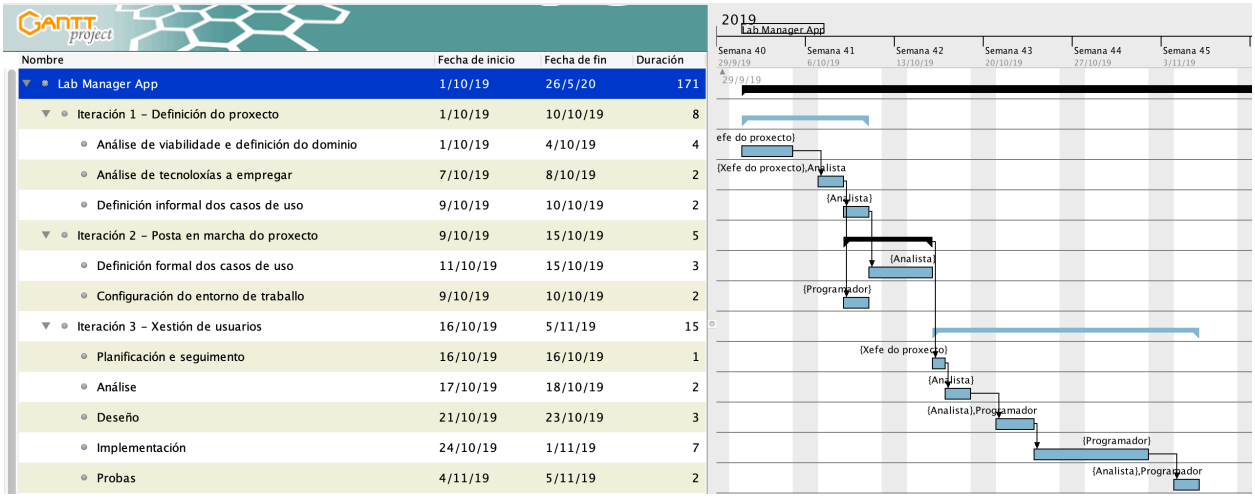
Como se observa na figura 5.1, exponse no diagrama de Gantt o resumo de cada iteración. Seguindo a planificación realizada, o desenvolvemento do proxecto supón un total de **171 días**, cunha media de 4 horas por día, de duración, cunha data de inicio do **1 de outubro de 2019** e unha data de finalización do **26 de maio do 2020**.



Nombre	Fecha de inicio	Fecha de fin	Duración
Lab Manager App	1/10/19	26/5/20	171
Iteración 1 – Definición do proxecto	1/10/19	10/10/19	8
Iteración 2 – Posta en marcha do proxecto	9/10/19	15/10/19	5
Iteración 3 – Xestión de usuarios	16/10/19	5/11/19	15
Iteración 4 – Xestión de laboratorios	6/11/19	25/11/19	14
Iteración 5 – Xestión de notificacións	26/11/19	11/12/19	12
Iteración 6 – Xestión de análises	12/12/19	31/12/19	14
Iteración 7 – Xestión de tarefas	1/1/20	15/1/20	11
Iteración 8 – Xestión de comentarios	16/1/20	29/1/20	10
Iteración 9 – Xestión de mostras	30/1/20	12/2/20	10
Iteración 10 – Integración con Google (Login e API Calendar)	13/2/20	25/3/20	30
Iteración 11 – Exportación de resultados en PDF	26/3/20	14/4/20	14
Iteración 12 – Elaboración da memoria	15/4/20	26/5/20	30

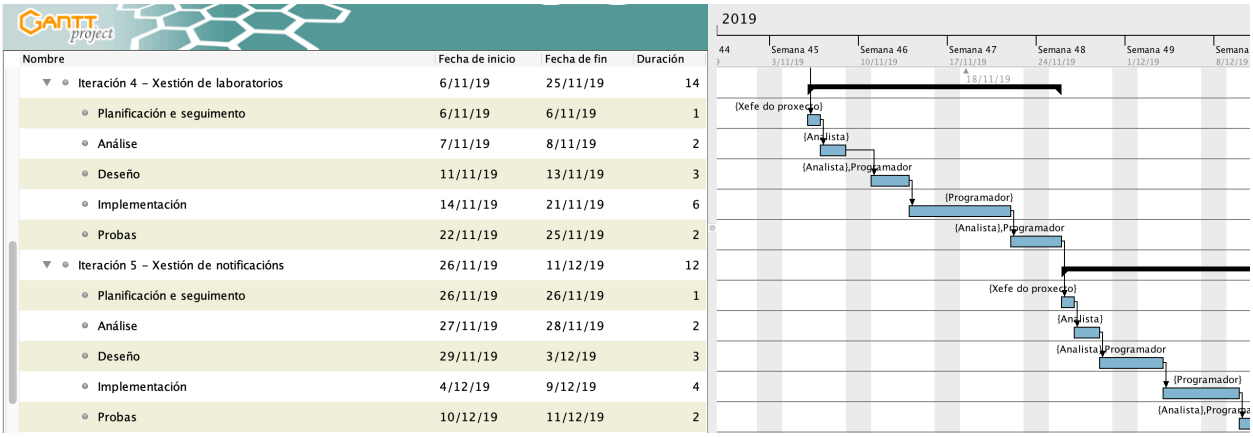
Figura 5.1: Diagrama de Gantt por iteracións

A continuación, móstranse as diferentes tarefas nas que se dividiron as iteracións:



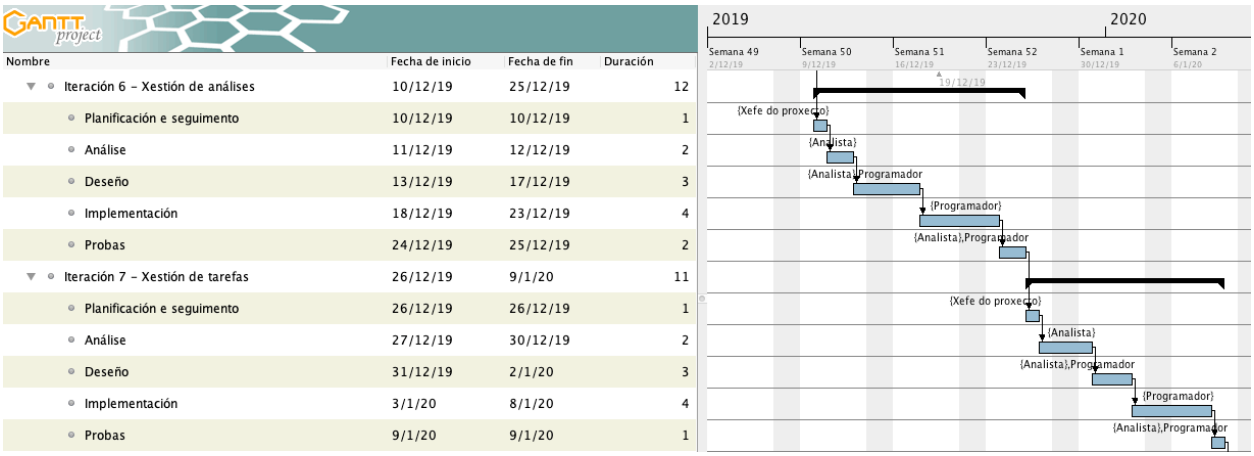
Nombre	Fecha de inicio	Fecha de fin	Duración
Lab Manager App	1/10/19	26/5/20	171
Iteración 1 – Definición do proxecto	1/10/19	10/10/19	8
Análise de viabilidade e definición do dominio	1/10/19	4/10/19	4
Análise de tecnoloxías a empregar	7/10/19	8/10/19	2
Definición informal dos casos de uso	9/10/19	10/10/19	2
Iteración 2 – Posta en marcha do proxecto	9/10/19	15/10/19	5
Definición formal dos casos de uso	11/10/19	15/10/19	3
Configuración do entorno de traballo	9/10/19	10/10/19	2
Iteración 3 – Xestión de usuarios	16/10/19	5/11/19	15
Planificación e seguimento	16/10/19	16/10/19	1
Análise	17/10/19	18/10/19	2
Deseño	21/10/19	23/10/19	3
Implementación	24/10/19	1/11/19	7
Probas	4/11/19	5/11/19	2

Figura 5.2: Diagrama de Gantt - tarefas por iteracións 1



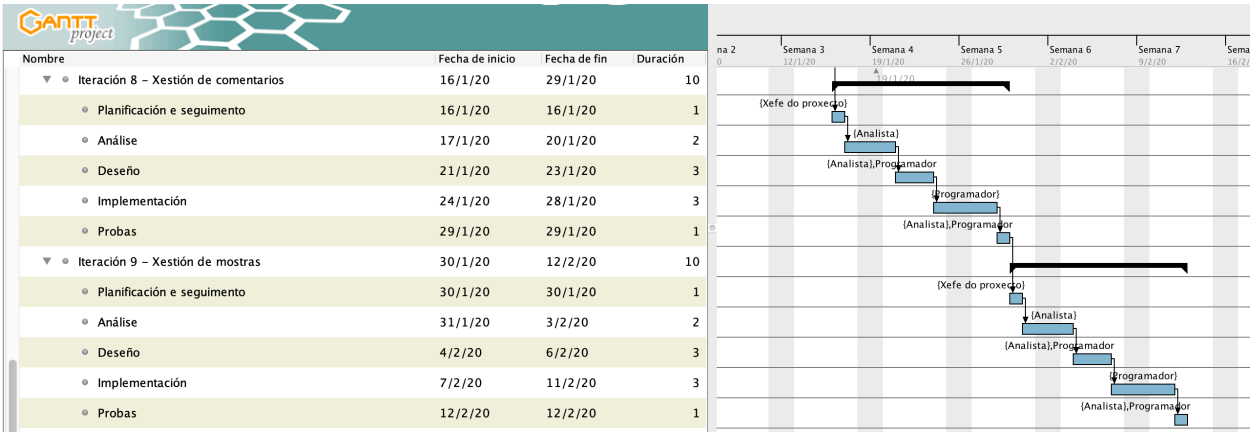
Nombre	Fecha de inicio	Fecha de fin	Duración
Iteración 4 – Xestión de laboratorios	6/11/19	25/11/19	14
Planificación e seguimento	6/11/19	6/11/19	1
Análise	7/11/19	8/11/19	2
Deseño	11/11/19	13/11/19	3
Implementación	14/11/19	21/11/19	6
Probas	22/11/19	25/11/19	2
Iteración 5 – Xestión de notificacións	26/11/19	11/12/19	12
Planificación e seguimento	26/11/19	26/11/19	1
Análise	27/11/19	28/11/19	2
Deseño	29/11/19	3/12/19	3
Implementación	4/12/19	9/12/19	4
Probas	10/12/19	11/12/19	2

Figura 5.3: Diagrama de Gantt - tarefas por iteracións 2



Nombre	Fecha de inicio	Fecha de fin	Duración
Iteración 6 – Xestión de análises	10/12/19	25/12/19	12
Planificación e seguimento	10/12/19	10/12/19	1
Análise	11/12/19	12/12/19	2
Deseño	13/12/19	17/12/19	3
Implementación	18/12/19	23/12/19	4
Probas	24/12/19	25/12/19	2
Iteración 7 – Xestión de tarefas	26/12/19	9/1/20	11
Planificación e seguimento	26/12/19	26/12/19	1
Análise	27/12/19	30/12/19	2
Deseño	31/12/19	2/1/20	3
Implementación	3/1/20	8/1/20	4
Probas	9/1/20	9/1/20	1

Figura 5.4: Diagrama de Gantt - tarefas por iteracións 3



Nombre	Fecha de inicio	Fecha de fin	Duración
Iteración 8 – Xestión de comentarios	16/1/20	29/1/20	10
Planificación e seguimento	16/1/20	16/1/20	1
Análise	17/1/20	20/1/20	2
Deseño	21/1/20	23/1/20	3
Implementación	24/1/20	28/1/20	3
Probas	29/1/20	29/1/20	1
Iteración 9 – Xestión de mostras	30/1/20	12/2/20	10
Planificación e seguimento	30/1/20	30/1/20	1
Análise	31/1/20	3/2/20	2
Deseño	4/2/20	6/2/20	3
Implementación	7/2/20	11/2/20	3
Probas	12/2/20	12/2/20	1

Figura 5.5: Diagrama de Gantt - tarefas por iteracións 4

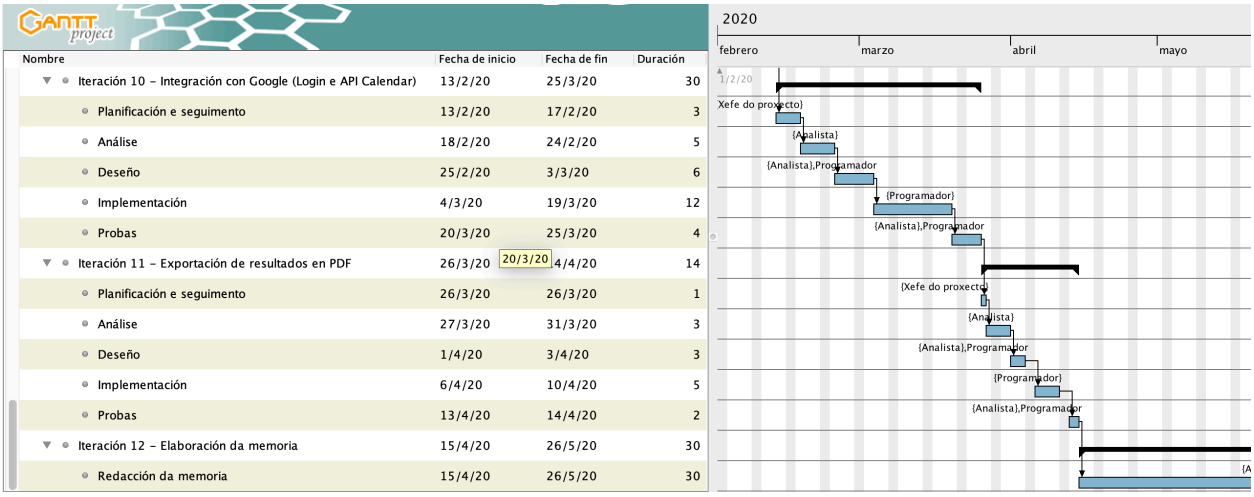


Figura 5.6: Diagrama de Gantt - tarefas por iteracións 5

Unha vez calculados os custos dos recursos humanos e técnicos e a estimación no tempo do proxecto, realízase o cálculo do custo estimado do proxecto.

Por unha parte obsérvase que a duración total do desenvolvemento completo do proxecto ascende a **892 horas**, polo que o custo de recursos humanos é:

- **Xefe do proxecto:** 68 horas que equivalen a *2.244€*.
- **Analista:** 428 horas que equivalen a *10.272€*.
- **Programador:** 396 horas que equivalen a *6.336€*.

Por último, é preciso calcular o custo dos recursos técnicos, que dividimos en dous bloques:

- **Recursos hardware:** Unicamente temos en conta o ordenador necesario para o desenvolvemento, cun valor de *2.000€*.
- **Recursos software:** Debido a que as ferramentas software empregadas son licenza libre, o custo é *0€*.

Con todo isto, móstrase na seguinte táboa o custo total estimado do proxecto:

Concepto	**Custo**
Recursos humanos	**18.852€**
Xefe do proxecto	2.244€
Analista	10.272€
Programador	6.336€
Recursos técnicos	**2.000€**
Software	0€
Hardware	2.000€
Subtotal	**20.852€**
IVA (21%)	4.378,92€
Total	**25.230,92€**

Capítulo 6

Requisitos do sistema

Neste capítulo explícanse os requisitos que debe cumprir o sistema final. Describiranse á súa vez os actores que interactuarán coa aplicación mediante as diferentes funcionalidades do sistema, detalladas no apartado de casos de uso.

6.1 Especificación dos requisitos

O obxectivo da especificación de requisitos software é o de expoñer de forma clara, concisa e sen ambigüidades as necesidades dos usuarios do sistema. Dividimos estes tipos de requisitos en dúas categorías: requirimentos funcionais e requirimentos non funcionais.

6.1.1 Requirimentos funcionais

Os requirimentos funcionais definen as funcións que realizará o sistema. Describen como os procesos do sistema transforman os datos de entrada en datos de saída e o procedemento que se leva a cabo para isto.

Os requirimentos funcionais explícanse en detalle, mediante os casos de uso, no apartado 6.3 (Casos de uso).

6.1.2 Requirimentos non funcionais

Por outra parte, os requirimentos non funcionais son aqueles que definen as características que limitan o sistema, como por exemplo o rendemento, fiabilidade, seguridade, etc.

Os requirimentos non funcionais que se determinan son os seguintes:

- **Seguridade**. A aplicación controla o acceso mediante a autenticación dos usuarios. Defínense regras dependendo do *endpoint REST* coas que se comproba, para cada ruta, se un usuario autenticado pode acceder.

Ao mesmo tempo, verifícase que os recursos propios dun usuario, como poden ser os seus comentarios ou datos persoais, só poidan ser recuperados polo propio usuario propietario deles.
Por último, defínense roles cos que se limitan as accións que pode realizar un usuario sobre a aplicación.

- **Usabilidade e simplicidade**. O uso da aplicación deséñase de forma sinxela e intuitiva. O sistema é SPA (*Single-Page Application*), de forma que a interface de usuario se mostre o máis sinxela posible.

- **Fiabilidade**. Os erros que se producen en tempo de execución son capturados e mostrados ao usuario cunha mensaxe "amigable".

- **Eficiencia**. As operacións do sistema están realizadas de maneira óptima, empregando a menor cantidade de recursos posibles, como memoria, conexións a base de datos, etc.

6.2 Actores

Existen tres tipos de actores dentro da aplicación (mostrados na figura 6.1):

- **Usuario non rexistrado**. É o usuario que accede a aplicación sen estar autenticado, polo que as únicas accións que pode realizar son as relacionadas co rexistro da aplicación.

- **Usuario empregado**. Correspóndese co rol *'EMPLOYEE'*. Estes usuarios poden realizar accións dentro dun laboratorio, como creación de análises, tarefas, comentarios, etc. Para isto deben estar asignados a un laboratorio, polo que tamén poderán buscar e solicitar asignación a estes.

- **Usuario *manager***. Correspóndese co rol *'LAB_MANAGER'*. Teñen acceso ás mesmas accións que os usuarios empregados pero, ademais, poden crear laboratorios aos cales se asignarán como responsables.

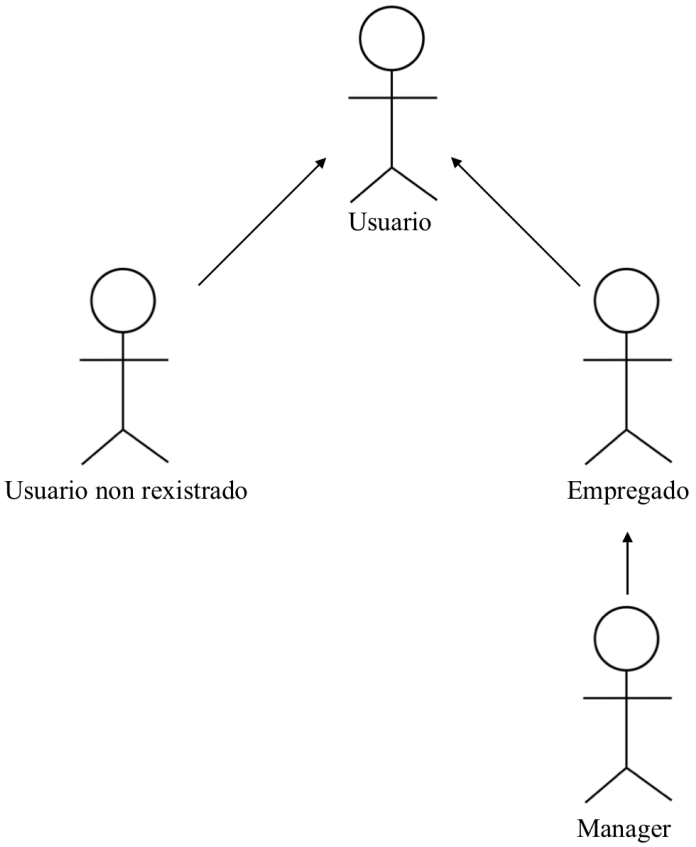


Figura 6.1: Actores

6.3 Casos de uso

6.3.1 Casos de uso de usuario

CU-101: Rexistrar usuario	
Descrición	Un novo usuario rexístrase no sistema.
Actores	Usuario non rexistrado.
Precondicións	O usuario non está rexistrado no sistema.
Fluxo básico	1. O usuario solicita ó sistema *logearse* mediante Google. 2. O sistema re-dirixe ó usuario á páxina de *login* de Google. 3. O usuario "loguéase" na páxina de Google. 4. O usuario é re-dirixido de novo ó sistema. 5. Móstraselle ó usuario os campos dun formulario con información adicional para o sistema. 6. O usuario cobre os campos adicionais. 7. O sistema valida os datos introducidos polo usuario e gárdaos na base de datos. 8. Re-diríxese ó usuario rexistrado á páxina de inicio do sistema.
Fluxo alternativo	1. O usuario introduce un dato non válido ou baleiro. 2. Infórmase ó usuario dos campos con erros e como solucionalos.
Postcondicións	Usuario rexistrado no sistema e autenticado na sesión actual.

Táboa 6.1: CU-101: Rexistrar usuario

CU-102: Iniciar sesión	
Descrición	Un usuario inicia sesión no sistema.
Actores	Usuario non rexistrado.
Precondicións	O usuario non está autenticado no sistema, pero está rexistrado.
Fluxo básico	1. O usuario solicita ó sistema autenticarse mediante Google. 2. O sistema re-dirixe ó usuario á páxina de *login* de Google. 3. O usuario autentícase na páxina de Google. 4. Re-diríxese ó usuario autenticado á páxina de inicio do sistema.
Fluxo alternativo	1. O usuario non se autentica na páxina de Google. 2. O sistema non autentica ó usuario.
Postcondicións	Usuario autenticado na sesión actual do sistema.

Táboa 6.2: CU-102: Iniciar sesión

CU-103: Pechar sesión	
Descrición	Un usuario pecha a sesión no sistema.
Actores	Usuario empregado, usuario *manager*.
Precondicións	O usuario está autenticado no sistema cunha sesión aberta.
Fluxo básico	1. O usuario solicita ó sistema pechar a sesión activa que ten nese momento. 2. O sistema elimina a sesión do usuario e re-diríxeo a páxina de inicio sen estar autenticado.
Fluxo alternativo	N/A
Postcondicións	Elimínase a sesión activa do usuario.

Táboa 6.3: CU-103: Pechar sesión

CU-104: Ver o perfil de usuario	
Descrición	Un usuario visualiza o seu perfil.
Actores	Usuario empregado, usuario *manager*.
Precondicións	O usuario está autenticado no sistema cunha sesión aberta.
Fluxo básico	1. O usuario solicita ó sistema visualizar os datos do seu perfil gardados no sistema. 2. O sistema re-dirixe ó usuario a páxina do seu perfil onde se visualizan os datos do usuario.
Fluxo alternativo	N/A
Postcondicións	Móstranse os datos do usuario.

Táboa 6.4: CU-104: Ver o perfil de usuario

CU-105: Modificar o perfil de usuario	
Descrición	Un usuario modifica os datos do seu perfil.
Actores	Usuario empregado, usuario *manager*.
Precondicións	O usuario está autenticado no sistema cunha sesión aberta.
Fluxo básico	1. O usuario solicita ó sistema visualizar os datos do seu perfil gardados no sistema. 2. O sistema re-dirixe ó usuario a páxina do seu perfil onde se visualizan os datos do usuario. 3. O usuario modifica os campos oportunos e solicita ó sistema actualizar os seus datos. 4. O sistema actualiza os datos gardándoos na base de datos. 5. O sistema informa ó usuario de que os cambios se gardaron correctamente.
Fluxo alternativo	1. O usuario introduce un dato non válido ou baleiro. 2. Infórmase ó usuario dos campos con erros e como solucionalos.
Postcondicións	Actualízanse os datos do usuario.

Táboa 6.5: CU-105: Modificar o perfil de usuario

CU-106: Ver o laboratorio do usuario	
Descrición	Un usuario visualiza os datos do seu laboratorio.
Actores	Usuario empregado, usuario *manager*.
Precondicións	O usuario está autenticado no sistema cunha sesión aberta e ten un laboratorio asignado.
Fluxo básico	1. O usuario solicita ó sistema acceder a información do seu laboratorio asignado. 2. O sistema re-dirixe ó usuario a páxina do laboratorio asignado onde se visualizan os datos do laboratorio e os usuario pertencentes ao mesmo.
Fluxo alternativo	N/A
Postcondicións	Móstranselle ao usuario os datos do laboratorio asignado.

Táboa 6.6: CU-106: Ver o laboratorio do usuario

CU-107: Ver todas as notificacións	
Descrición	Un usuario visualiza todas as súas notificacións.
Actores	Usuario empregado, usuario *manager*.
Precondicións	O usuario está autenticado.
Fluxo básico	1. O usuario solicita ó sistema visualizar todas as súas notificacións. 2. O sistema re-dirixe ó usuario a páxina de notificacións, onde se visualizan todas as notificacións que ten o usuario.
Fluxo alternativo	1. O usuario non ten ningunha notificación e solicita ó sistema visualizar todas as súas notificacións. 2. O sistema re-dirixe ó usuario a páxina de notificacións e informa ó usuario de que non ten ningunha notificación.
Postcondicións	Móstranselle as notificacións ao usuario.

Táboa 6.7: CU-107: Ver todas as notificacións

CU-108: Acceder a unha notificación	
Descrición	Un usuario accede a unha notificación.
Actores	Usuario empregado, usuario *manager*.
Precondicións	O usuario está autenticado e ten como mínimo unha notificación.
Fluxo básico	1. O usuario solicita ó sistema visualizar todas as súas notificacións. 2. O sistema re-dirixe ó usuario á páxina de notificacións, onde se visualizan todas as notificacións que ten o usuario. 3. O usuario solicita acceder a unha notificación mostradas polo sistema. 4. O sistema móstralle os datos da notificación ó usuario.
Fluxo alternativo	N/A
Postcondicións	Móstraselle a notificación ao usuario.

Táboa 6.8: CU-108: Acceder a unha notificación

CU-109: Eliminar perfil de usuario	
Descrición	Un usuario elimina o seu perfil.
Actores	Usuario empregado, usuario *manager*.
Precondicións	O usuario está autenticado no sistema cunha sesión aberta.
Fluxo básico	1. O usuario solicita ó sistema eliminar o seu perfil do sistema. 2. O sistema solicita ó usuario confirmación para eliminar o seu perfil. 3. O usuario confirma a eliminación do seu perfil. 4. O sistema modifica os seguintes datos da base de datos: Cambia a columna *active* a falso e elimina os valores de *user_nick_name*, *roles*, *user_position*, *laboratory_id* e *laboratory_id_waiting_for_assign*
Fluxo alternativo	Fluxo alternativo 1 (usuario *manager* con un laboratorio asignado sen máis usuarios que o propio *manager*): 1. O sistema elimina os datos citados no fluxo básico e, ademais, elimina de base de datos o laboratorio que pertencía ó usuario. Fluxo alternativo 2 (usuario *manager* con un laboratorio asignado con máis usuarios asignados ó laboratorio): 1. Antes de eliminar os datos mostrados no paso 4 do fluxo básico, o sistema re-dirixe ó usuario a unha páxina na que se mostran os usuarios do laboratorio. Móstrase unha mensaxe indicando que o usuario *manager* debe seleccionar outro usuario do laboratorio para convertelo no encargado do laboratorio. 2. O usuario selecciona un usuario. 3. O sistema solicita confirmación ó usuario de se está seguro de asignar dito usuario como encargado. 4. O usuario confirma a asignación. 5. O sistema elimina os datos, como no paso 4 do fluxo básico, e, aparte, asigna o usuario seleccionado como encargado do laboratorio. En caso de que o usuario seleccionado teña rol *'EMPLOYEE'* actualízase a rol *'LAB_manager'*.
Postcondicións	Elimínase o usuario.

Táboa 6.9: CU-109: Eliminar perfil de usuario

6.3.2 Casos de uso de laboratorio

CU-201: Crear laboratorio	
Descrición	Un usuario *manager* crea un novo laboratorio.
Actores	Usuario *manager*.
Precondicións	O usuario está autenticado no sistema cunha sesión aberta e non ten un laboratorio asignado.
Fluxo básico	1. O usuario solicita ó sistema comezar o proceso de creación de laboratorio. 2. O sistema mostra uns campos dun formulario de creación de laboratorio. 3. O usuario cobre os campos necesarios e solicita a creación. 4. O sistema garda en base de datos o novo laboratorio e móstralle ó usuario os novos datos do laboratorio creado.
Fluxo alternativo	1. O usuario introduce un dato non válido ou baleiro. 2. Infórmase ó usuario dos campos con erros e como solucionalos.
Postcondicións	Créase un novo laboratorio e asígnaselle ó usuario.

Táboa 6.10: CU-201: Crear laboratorio

CU-202: Buscar laboratorio	
Descrición	Un usuario busca un laboratorio.
Actores	Usuario *manager* e usuario empregado.
Precondicións	O usuario está autenticado no sistema cunha sesión aberta e non ten un laboratorio asignado.
Fluxo básico	1. O usuario solicita ó sistema o proceso de procura de laboratorios. 2. O sistema mostra un campo dun formulario onde se pode introducir o nome do laboratorio que se desexa buscar. 3. O usuario cobre os campos necesarios. 4. O sistema re-dirixe ó usuario a páxina na que se mostran os laboratorios que cumpran co criterio de procura.
Fluxo alternativo	1. O usuario introduce un dato non válido ou baleiro. 2. Infórmase ó usuario dos campos con erros e como solucionalos.
Postcondicións	Móstranse os laboratorio que coinciden co criterio de procura.

Táboa 6.11: CU-202: Buscar laboratorio

CU-203: Solicitar asignación a un laboratorio	
Descrición	Un usuario solicita ser asignado a un laboratorio.
Actores	Usuario *manager* e usuario empregado.
Precondicións	O usuario está autenticado no sistema cunha sesión aberta e non ten un laboratorio asignado.
Fluxo básico	1. O usuario busca un laboratorio mediante o caso de uso *CU-202: Buscar laboratorio.* 2. O usuario solicita unirse ó laboratorio seleccionado. 3. O sistema envía un correo electrónico e unha notificación ó usuario encargado do laboratorio conforme un usuario solicitou unirse ao mesmo. O usuario é re-dirixido a páxina do laboratorio, na que se mostra unha mensaxe de que a súa petición foi enviada e está á espera de ser aceptada polo encargado do laboratorio.
Fluxo alternativo	N/A
Postcondicións	Envíase unha notificación de asignación do usuario.

Táboa 6.12: CU-203: Solicitar asignación a un laboratorio

CU-204: Aceptar/rexeitar asignación a un laboratorio	
Descrición	Un usuario *manager* acepta/rexeita a petición de ser asignado a un laboratorio dun usuario.
Actores	Usuario *manager*.
Precondicións	O usuario está autenticado no sistema e ten unha notificación de solicitude de asignación.
Fluxo básico	1. O usuario accede mediante o caso de uso *CU-108: Acceder a unha notificación.* 2. O usuario solicita a aceptación da petición. 3. O sistema envía un correo electrónico e unha notificación ó usuario solicitante e asígnao ao laboratorio solicitado.
Fluxo alternativo	1. O usuario accede mediante o caso de uso *CU-108: Acceder a unha notificación.* 2. O usuario solicita rexeitar a petición de asignación. 3. O sistema envía un correo electrónico e unha notificación ó usuario solicitante e elimina a petición de asignación.
Postcondicións	Envíase unha notificación de asignación do usuario.

Táboa 6.13: CU-204: Aceptar/rexeitar asignación a un laboratorio

CU-205: Editar laboratorio	
Descrición	Un usuario edita un laboratorio.
Actores	Usuario *manager*.
Precondicións	1. O usuario está autenticado no sistema. 2. Ten un laboratorio asignado. 3. É o *manager* do laboratorio.
Fluxo básico	1. O usuario solicita ó sistema acceder a información do seu laboratorio. 2. O sistema mostra a información do laboratorio que ten asignado o usuario. 3. O usuario solicita actualizar dito laboratorio. 4. O sistema re-dirixe ó usuario a unha nova páxina con un formulario de edición de laboratorio. 5. O usuario cobre os campos necesarios. 6. O sistema garda os novos datos e informa ó usuario de que a actualización se realizou correctamente.
Fluxo alternativo	1. O usuario introduce un dato non válido ou baleiro. 2. Infórmase ó usuario dos campos con erros e como solucionalos.
Postcondicións	Móstrase a nova información actualizada do laboratorio.

Táboa 6.14: CU-205: Editar laboratorio

6.3.3 Casos de uso de análises

CU-301: Crear análise	
Descrición	Un usuario crea unha nova análise.
Actores	Usuario *manager* e usuario empregado.
Precondicións	O usuario está autenticado no sistema e ten un laboratorio asignado.
Fluxo básico	1. O usuario solicita ó sistema acceder ó procedemento de creación de análise. 2. O sistema re-dirixe ó usuario a unha páxina na que mostran uns campos dun formulario de creación de análises. 3. O usuario cobre os campos necesarios. 4. O sistema re-dirixe ó usuario a páxina de información da análise mostrando o resumo de dita análise.
Fluxo alternativo	1. O usuario introduce un dato non válido ou baleiro. 2. Infórmase ó usuario dos campos con erros e como solucionalos.
Postcondicións	Créase unha nova análise.

Táboa 6.15: CU-301: Crear análise

CU-302: Buscar todas as análises	
Descrición	Un usuario busca todas as súas análises.
Actores	Usuario *manager* e usuario empregado.
Precondicións	O usuario está autenticado no sistema.
Fluxo básico	1. O usuario solicita ó sistema a procura de todas as súas análises. 2. O sistema re-dirixe ó usuario a unha nova páxina na que se mostran todas as análises do usuario ou unha mensaxe indicando que o usuario non ten ningunha análise, no caso que este non teña análises rexistradas.
Fluxo alternativo	N/A
Postcondicións	Móstranse as análises do usuario.

Táboa 6.16: CU-302: Buscar todas as análises

CU-303: Buscar unha análise	
Descrición	Un usuario busca unha análise.
Actores	Usuario *manager* e usuario empregado.
Precondicións	O usuario está autenticado no sistema e ten como mínimo unha análise rexistrada.
Fluxo básico	1. O usuario busca análises mediante o caso de uso *CU-302: Buscar todas as análises*. 2. O sistema mostra as análises do usuario. 3. O usuario solicita ó sistema mostrar unha análise en base a criterios de procura. 4. O sistema actualiza a táboa de resultados en base ó criterio inserido polo usuario.
Fluxo alternativo	N/A
Postcondicións	Móstraselle a análise ao usuario.

Táboa 6.17: CU-303: Buscar unha análise

CU-304: Acceder a unha análise	
Descrición	Un usuario accede a unha análise.
Actores	Usuario *manager* e usuario empregado.
Precondicións	O usuario está autenticado no sistema e ten como mínimo unha análise rexistrada.
Fluxo básico	1. O usuario busca análises mediante o caso de uso *CU-302: Buscar todas as análises* ou *CU-303: Buscar unha análise*. 2. O usuario selecciona a análise á que desexa acceder. 3. O sistema re-dirixe ó usuario a páxina da análise seleccionada, mostrando a información de dita análise.
Fluxo alternativo	N/A
Postcondicións	O usuario accede á análise seleccionada.

Táboa 6.18: CU-304: Acceder a unha análise

CU-305: Editar análise	
Descrición	Un usuario edita unha análise.
Actores	Usuario *manager* e usuario empregado.
Precondicións	O usuario está autenticado no sistema e ten como mínimo unha análise rexistrada.
Fluxo básico	1. O usuario accede a unha análise mediante o caso de uso *CU-304: Acceder a unha análise.* 2. O usuario solicita ó sistema o procedemento de edición da análise. 3. O sistema mostra os datos actuais da análise coa posibilidade de modificalos. 4. O usuario modifica os datos desexados. 5. O sistema garda a información actualizada e informa ó usuario de que a información se actualizou correctamente.
Fluxo alternativo	1. O usuario introduce un dato non válido ou baleiro. 2. Infórmase ó usuario dos campos con erros e como solucionalos.
Postcondicións	A información da análise actualízase correctamente.

Táboa 6.19: CU-305: Editar análise

CU-306: Engadir resultado a unha análise	
Descrición	Un usuario engade un resultado a unha análise.
Actores	Usuario *manager* e usuario empregado.
Precondicións	O usuario está autenticado no sistema e ten como mínimo unha análise rexistrada.
Fluxo básico	1. O usuario accede a unha análise mediante o caso de uso *CU-304: Acceder a unha análise.* 2. O usuario solicita ó sistema engadir un resultado á análise. 3. O sistema mostra diferentes resultados a engadir. 4. O usuario selecciona un dos resultados. 5. O sistema garda a información actualizada informa ó usuario de que o resultado se engadiu correctamente.
Fluxo alternativo	N/A
Postcondicións	O resultado da análise engádese correctamente.

Táboa 6.20: CU-306: Engadir resultado a unha análise

CU-307: Pechar análise	
Descrición	Un usuario pecha unha análise.
Actores	Usuario *manager* e usuario empregado.
Precondicións	1. O usuario está autenticado no sistema. 2. Ten como mínimo unha análise rexistrada. 3. A análise ten todas as tarefas en estado pechado.
Fluxo básico	1. O usuario accede a unha análise mediante o caso de uso *CU-304: Acceder a unha análise.* 2. O usuario solicita ó sistema pechar a análise. 3. O sistema garda o novo estado da análise e móstralle ao usuario unha mensaxe indicando que a análise se pechou correctamente.
Fluxo alternativo	1. A análise non ten todas as tarefas pechadas e o usuario solicita pechar a análise. 2. O sistema mostra un erro indicando que non se permite pechar unha análise que non ten todas as tarefas pechadas.
Postcondicións	A análise péchase correctamente.

Táboa 6.21: CU-307: Pechar análise

CU-308: Reabrir análise	
Descrición	Un usuario reabre unha análise.
Actores	Usuario *manager* e usuario empregado.
Precondicións	1. O usuario está autenticado no sistema. 2. Ten como mínimo unha análise rexistrada. 3. A análise está en estado pechado.
Fluxo básico	1. O usuario accede a unha análise mediante o caso de uso *CU-304: Acceder a unha análise.* 2. O usuario solicita ó sistema reabrir a análise. 3. O sistema garda o novo estado da análise e móstralle ao usuario unha mensaxe indicando que a análise se reabriu correctamente.
Fluxo alternativo	N/A
Postcondicións	A análise reábrese correctamente.

Táboa 6.22: CU-308: Reabrir análise

CU-309: Exportar PDF da análise	
Descrición	Un usuario exporta unha análise en PDF.
Actores	Usuario *manager* e usuario empregado.
Precondicións	1. O usuario está autenticado no sistema. 2. Ten como mínimo unha análise rexistrada. 3. A análise a exportar ten estado pechado. 4. A análise ten tarefas en estado pechado. 5. A análise ten un resultado.
Fluxo básico	1. O usuario accede a unha análise mediante o caso de uso *CU-304: Acceder a unha análise.* 2. O usuario solicita ó sistema exportar a análise en formato PDF. 3. O sistema descarga un arquivo en formato PDF co resumo da análise, coas súas tarefas, mostras e resultados.
Fluxo alternativo	N/A
Postcondicións	Expórtase a análise en formato PDF.

Táboa 6.23: CU-309: Exportar PDF da análise

CU-310: Eliminar análise	
Descrición	Un usuario elimina unha análise.
Actores	Usuario *manager* e usuario empregado.
Precondicións	1. O usuario está autenticado no sistema. 2. Ten como mínimo unha análise rexistrada. 3. A análise a eliminar non ten tarefas ou tenas en estado cancelado.
Fluxo básico	1. O usuario accede a unha análise mediante o caso de uso *CU-304: Acceder a unha análise.* 2. O usuario solicita ó sistema o procedemento de eliminación da análise. 3. O sistema mostra unha mensaxe de confirmación de borrado. 4. O usuario confirma a eliminación. 5. O sistema elimina a análise e re-dirixe ó usuario a páxina da listaxe de análises, informando ó usuario de que a análise se eliminou correctamente.
Fluxo alternativo	1. O usuario non confirma o borrado da análise. 2. O sistema non elimina dita análise.
Postcondicións	Elimínase a análise.

Táboa 6.24: CU-310: Eliminar análise

6.3.4 Casos de uso de tarefas

CU-401: Crear tarefa	
Descrición	Un usuario crea unha nova tarefa.
Actores	Usuario *manager* e usuario empregado.
Precondicións	1. O usuario está autenticado no sistema. 2. O usuario ten un laboratorio asignado. 3. O usuario ten, como mínimo, unha análise.
Fluxo básico	1. O usuario accede a unha análise mediante o caso de uso *CU-304: Acceder a unha análise.* 2. O usuario solicita ó sistema crear unha tarefa sobre a análise a que accedeu. 3. O sistema re-dirixe ó usuario a unha páxina na que mostran uns campos dun formulario de creación de tarefas. 4. O usuario cobre os campos necesarios. 5. O sistema re-dirixe ó usuario a páxina de información da análise coa nova tarefa na lista de tarefas e mostra unha mensaxe indicando que a tarefa se creou correctamente.
Fluxo alternativo	1. O usuario introduce un dato non válido ou baleiro. 2. Infórmase ó usuario dos campos con erros e como solucionalos.
Postcondicións	Créase unha nova tarefa.

Táboa 6.25: CU-401: Crear tarefa

CU-402: Acceder a unha tarefa	
Descrición	Un usuario accede a unha tarefa.
Actores	Usuario *manager* e usuario empregado.
Precondicións	1. O usuario está autenticado no sistema. 2. O usuario ten un laboratorio asignado. 3. O usuario ten, como mínimo, unha tarefa.
Fluxo básico	1. O usuario accede a unha análise mediante o caso de uso *CU-304: Acceder a unha análise*. 2. O usuario solicita ó sistema acceder a unha das tarefas que se mostran na lista de tarefas da análise. 3. O sistema re-dirixe ó usuario a páxina de información da tarefa na que se mostras os detalles da mesma, xunto cos comentarios e mostras que contén.
Fluxo alternativo	N/A
Postcondicións	Accédese a tarefa.

Táboa 6.26: CU-402: Acceder a unha tarefa

CU-403: Editar unha tarefa	
Descrición	Un usuario edita unha tarefa.
Actores	Usuario *manager* e usuario empregado.
Precondicións	1. O usuario está autenticado no sistema. 2. O usuario ten un laboratorio asignado. 3. O usuario ten, como mínimo, unha tarefa.
Fluxo básico	1. O usuario accede a unha tarefa mediante o caso de uso *CU-402: Acceder a unha tarefa*. 2. O usuario solicita ó sistema editar a tarefa. 3. O sistema mostra uns campos dun formulario cos datos da tarefa coa posibilidade de modificalos. 4. O usuario cobre os campos que desexa modificar. 5. O sistema garda os datos da tarefa e mostra unha mensaxe indicando que se actualizou correctamente.
Fluxo alternativo	1. O usuario introduce un dato non válido ou baleiro. 2. Infórmase ó usuario dos campos con erros e como solucionalos.
Postcondicións	Actualízase a tarefa.

Táboa 6.27: CU-403: Editar unha tarefa

CU-404: Cambiar estado dunha tarefa	
Descrición	Un usuario cambia o estado dunha tarefa.
Actores	Usuario *manager* e usuario empregado.
Precondicións	1. O usuario está autenticado no sistema. 2. O usuario ten un laboratorio asignado. 3. O usuario ten, como mínimo, unha tarefa.
Fluxo básico	1. O usuario accede a unha tarefa mediante o caso de uso *CU-402: Acceder a unha tarefa.* 2. O usuario solicita ó sistema o procedemento para cambiar o estado da tarefa á que accedeu. 3. O sistema mostra os posibles estados que pode ter a tarefa. 4. O usuario selecciona un estado. 5. O sistema garda o novo estado da tarefa e mostra unha mensaxe indicando que se actualizou correctamente.
Fluxo alternativo	1. O usuario non selecciona ningún estado. 2. O sistema non cambia o estado da tarefa.
Postcondicións	Actualízase o estado da tarefa.

Táboa 6.28: CU-404: Cambiar estado dunha tarefa

CU-405: Engadir data de finalización dunha tarefa	
Descrición	Un usuario engade a data de finalización a unha tarefa.
Actores	Usuario *manager* e usuario empregado.
Precondicións	1. O usuario está autenticado no sistema. 2. O usuario ten un laboratorio asignado. 3. O usuario ten, como mínimo, unha tarefa. 4. A data de finalización debe ser posterior á data actual.
Fluxo básico	1. O usuario accede a unha tarefa mediante o caso de uso *CU-402: Acceder a unha tarefa.* 2. O usuario solicita ó sistema engadir a data de finalización da tarefa. 3. O sistema abre un formulario no que se mostran un calendario e unha mensaxe. 4. O usuario selecciona unha data e insire unha mensaxe. 5. O sistema comunícase coa API Calendar de Google, engadindo un novo evento no calendario do usuario coa data seleccionada, a mensaxe e valores por defecto. 6. O sistema garda a data de finalización e mostra unha mensaxe indicando que a data de finalización se gardou correctamente.
Fluxo alternativo	1. O usuario introduce un dato non válido ou baleiro. 2. Infórmase ó usuario dos campos con erros e como solucionalos.
Postcondicións	Gárdase a data de finalización da tarefa e engádese un novo evento no calendario do usuario.

Táboa 6.29: CU-405: Engadir data de finalización dunha tarefa

CU-406: Engadir comentario nunha tarefa	
Descrición	Un usuario engade un comentario nunha tarefa.
Actores	Usuario *manager* e usuario empregado.
Precondicións	1. O usuario está autenticado no sistema. 2. O usuario ten un laboratorio asignado. 3. O usuario ten, como mínimo, unha tarefa.
Fluxo básico	1. O usuario accede a unha tarefa mediante o caso de uso *CU-402: Acceder a unha tarefa.* 2. O usuario solicita ó sistema engadir un comentario sobre a tarefa. 3. O sistema abre un formulario no que mostra un campo para inserir o comentario. 4. O sistema garda o comentario e mostra unha mensaxe indicando que o comentario se gardou correctamente.
Fluxo alternativo	1. O usuario introduce un dato non válido ou baleiro. 2. Infórmase ó usuario dos campos con erros e como solucionalos.
Postcondicións	Gárdase o comentario.

Táboa 6.30: CU-406: Engadir comentario nunha tarefa

CU-407: Eliminar comentario nunha tarefa	
Descrición	Un usuario elimina un comentario nunha tarefa.
Actores	Usuario *manager* e usuario empregado.
Precondicións	1. O usuario está autenticado no sistema. 2. O usuario ten un laboratorio asignado. 3. O usuario ten, como mínimo, unha tarefa. 4. O comentario a eliminar foi engadido polo mesmo usuario.
Fluxo básico	1. O usuario accede a unha tarefa mediante o caso de uso *CU-402: Acceder a unha tarefa.* 2. O usuario solicita ó sistema o procedemento de eliminación sobre o comentario que desexa. 3. O sistema mostra unha mensaxe de confirmación indicando se o usuario está seguro de eliminar o comentario. 4. O usuario confirma a eliminación. 5. O sistema elimina o comentario do sistema e mostra unha mensaxe indicando que o comentario se eliminou correctamente.
Fluxo alternativo	1. O usuario cancela a confirmación do borrado do comentario. 2. O sistema non elimina dito comentario.
Postcondicións	Elimínase o comentario.

Táboa 6.31: CU-407: Eliminar comentario nunha tarefa

CU-408: Eliminar tarefa	
Descrición	Un usuario elimina unha tarefa.
Actores	Usuario *manager* e usuario empregado.
Precondicións	1. O usuario está autenticado no sistema. 2. O usuario ten un laboratorio asignado. 3. O usuario ten, como mínimo, unha análise. 4. A tarefa ten estado cancelado.
Fluxo básico	1. O usuario accede a unha tarefa mediante o caso de uso *CU-402: Acceder a unha tarefa.* 2. O usuario solicita ó sistema eliminar a tarefa á que accedeu. 3. O sistema mostra un mensaxe indicando se o usuario está seguro de eliminar a tarefa. 4. O usuario confirma a eliminación. 5. O sistema elimina a tarefa e re-dirixe ó usuario á páxina da análise á que pertencía a tarefa, mostrando unha mensaxe de que a tarefa se eliminou correctamente.
Fluxo alternativo	1. O usuario cancela a confirmación de eliminación. 2. O sistema non elimina a tarefa.
Postcondicións	Eliminase a tarefa.

Táboa 6.32: CU-408: Eliminar tarefa

CU-409: Visualizar tarefas abertas	
Descrición	Un usuario visualiza as súas tarefas abertas.
Actores	Usuario *manager* e usuario empregado.
Precondicións	O usuario está autenticado no sistema.
Fluxo básico	1. O usuario solicita ó sistema visualizar as súas tarefas en estado aberto. 2. O sistema re-dirixe ó usuario a unha nova páxina na que se mostran todas as tarefas en estado *OPEN* e *IN_PROGRESS* que ten o usuario.
Fluxo alternativo	1. O usuario accede as súas tarefas abertas, pero non dispón de ningunha. 2. O sistema mostra unha mensaxe na nova páxina indicando de que o usuario non ten ningunha tarefa aberta.
Postcondicións	Móstranse as tarefas abertas do usuario.

Táboa 6.33: CU-409: Visualizar tarefas aberta

6.3.5 Casos de uso de mostras

CU-501: Engadir mostra	
Descrición	Un usuario engade unha nova mostra.
Actores	Usuario *manager* e usuario empregado.
Precondicións	1. O usuario está autenticado no sistema. 2. O usuario ten un laboratorio asignado. 3. O usuario ten, como mínimo, unha tarefa.
Fluxo básico	1. O usuario accede a unha tarefa mediante o caso de uso *CU-402: Acceder a unha tarefa.* 2. O usuario solicita ó sistema engadir unha nova mostra sobre a tarefa procurada. 3. O sistema re-dirixe ó usuario a unha nova páxina na que se mostran campos dun formulario de creación de mostras. 4. O usuario cobre os campos. 5. O sistema garda a nova mostra e informa ó usuario de que a mostra se creou correctamente.
Fluxo alternativo	1. O usuario introduce un dato non válido ou baleiro. 2. Infórmase ó usuario dos campos con erros e como solucionalos.
Postcondicións	Créase a mostra.

Táboa 6.34: CU-501: Engadir mostra

CU-502: Engadir resultado a unha mostra	
Descrición	Un usuario engade un resultado a unha mostra.
Actores	Usuario *manager* e usuario empregado.
Precondicións	1. O usuario está autenticado no sistema. 2. O usuario ten un laboratorio asignado. 3. O usuario ten, como mínimo, unha tarefa. 4. A tarefa ten, como mínimo, unha mostra.
Fluxo básico	1. O usuario accede a unha tarefa mediante o caso de uso *CU-402: Acceder a unha tarefa.* 2. O usuario solicita ó sistema engadir un resultado na mostra que desexa. 3. O sistema mostra un formulario que contén un campo de resultado. 4. O usuario cobre o campo do resultado. 5. O sistema garda o resultado e mostra unha mensaxe que indica que o resultado se gardou correctamente.
Fluxo alternativo	1. O usuario introduce un dato non válido ou baleiro. 2. Infórmase ó usuario dos campos con erros e como solucionalos.
Postcondicións	Engádese resultado á mostra.

Táboa 6.35: CU-502: Engadir resultado a unha mostra

CU-503: Eliminar mostra de unha tarefa	
Descrición	Un usuario elimina unha mostra de unha tarefa.
Actores	Usuario *manager* e usuario empregado.
Precondicións	1. O usuario está autenticado no sistema. 2. O usuario ten un laboratorio asignado. 3. O usuario ten, como mínimo, unha tarefa. 4. A tarefa ten polo menos unha mostra.
Fluxo básico	1. O usuario accede a unha tarefa mediante o caso de uso *CU-402: Acceder a unha tarefa.* 2. O usuario solicita ó sistema eliminar a mostra que desexa. 3. O sistema mostra unha mensaxe indicando se o usuario está seguro de eliminar a mostra. 4. O usuario confirma a eliminación da mostra. 5. O sistema elimina a mostra do sistema e mostra unha mensaxe indicando que a mostra se eliminou correctamente.
Fluxo alternativo	Fluxo alternativo 1: 1. O usuario non confirma o borrado da mostra. 2. O sistema non elimina a mostra.
Postcondicións	Elimínase a mostra.

Táboa 6.36: CU-503: Eliminar mostra de unha tarefa

CU-504: Editar mostra	
Descrición	Un usuario edita unha mostra.
Actores	Usuario *manager* e usuario empregado.
Precondicións	1. O usuario está autenticado no sistema. 2. O usuario ten un laboratorio asignado. 3. O usuario ten, como mínimo, unha tarefa. 4. A tarefa ten polo menos unha mostra.
Fluxo básico	1. O usuario accede a unha tarefa mediante o caso de uso *CU-402: Acceder a unha tarefa.* 2. O usuario solicita ó sistema editar a mostra que desexa. 3. O sistema mostra un formulario para actualizar os datos da mostra. 4. O usuario cobre os campos do formulario. 5. O sistema actualiza os datos da mostra unha mensaxe indicando que a mostra se actualizou correctamente.
Fluxo alternativo	1. O usuario non confirma o envío do formulario de actualización da mostra. 2. O sistema non actualiza a mostra.
Postcondicións	Actualízase a mostra.

Táboa 6.37: CU-504: Editar mostra

Capítulo 7

Deseño da aplicación

Neste capítulo expóñense os aspectos relacionados co deseño da aplicación de cara a construcción da mesma, tendo en conta a arquitectura.

Para o desenvolvemento dos requisitos expostos no apartado anterior (6.1) lévase a cabo a utilización de diferentes patróns de deseño e boas prácticas.

A continuación, detallaranse os patróns de deseño e a arquitectura do sistema.

7.1 Patróns de deseño

7.1.1 Model-View-Controller

O Modelo-Vista-Controlador (MVC) é un patrón de deseño amplamente empregado na arquitectura de aplicacións web. Contén tres capas principais, modelo, vista e controlador. Esta arquitectura é a empregada neste proxecto, como se pode ver no apartado 4.1.

O patrón de MVC era orixinalmente aplicado no entorno de programación Smalltalk, desenvolvido en Xerox PARC (Goldberg e Robson, 1985), como unha forma de estruturar aplicacións iterativas modulares. A evolución de dito patrón difire moito da implementación orixinal. A continuación, detállase a interpretación do patrón dentro das aplicacións web.

O compoñente **modelo** encapsula a funcionalidade da lóxica de negocio da aplicación. Isto inclúe o estado da aplicación e as operacións que poden cambiar dito estado. Responde as peticións que chegan a través do controlador realizando diferentes accións sobre elas. Outros dos puntos importantes do modelo son manter a base de datos realizando operacións sobre ela ou a iteración con servizos externos como un servizo web.

O compoñente **vista** ten a función principal de mostrarlle ó usuario información de forma gráfica, a través da interface gráfica de usuario. A través de dita interface, o usuario pode

interactuar realizando operacións para recuperar información do modelo mediante o controlador. Estas operacións lévanse a cabo mediante botóns e formularios da vista.

O compoñente **controlador** actúa como ”intermediario”, é dicir, encárgase de xestionar as peticións que realiza o usuario a través da vista e comunícase co modelo para levar a cabo ditas peticións. Tamén é o encargado de devolver a información recuperada no modelo á vista.

A principal vantaxe do patrón Modelo-Vista-Controlador é o desacople entre os principais compoñentes, facendo sinxela a re-utilización e mantebilidade do código. Favorece á súa vez o desenvolvemento do código en paralelo, debido a que varios desenvolvedores poden traballar en diferentes compoñentes á vez.

7.1.2 Repository

A función do repositorio [22] é separar a capa lóxica que recupera os datos da capa de lóxica de negocio. Á lóxica de negocio débelle ser indiferente o tipo de datos que se manexan na capa de datos, por exemplo, a capa lóxica pode ser unha base de datos, un servizo web, etc.

O repositorio actúa como mediador entre a capa de acceso a datos e a implementación da lóxica de negocio. *Mapea* os datos da capa lóxica ás entidades de negocio e, pola contra, persiste os cambios que se realicen sobre as entidades. Esta separación ten unha serie de beneficios, entre outros, centralizar a lóxica de acceso a datos nun só punto, evitando desa maneira posibles erros, ou prover unha arquitectura flexible que permite ser adaptada aos posibles cambios que se necesiten ao longo do tempo.

Este patrón emprégase na capa de acceso a datos, que se explica no apartado 7.2.1 onde se definen os repositorios da aplicación.

7.1.3 Facade

O patrón de deseño *Facade* [23] é un patrón estrutural amplamente empregado na programación orientada a obxectos. Proporciona unha interface unificada a un conxunto de interfaces nun subsistema. A fachada define unha interface de alto nivel.

Grazas ó anteriormente citado, algúns dos beneficios do patrón fachada son conseguir minimizar as dependencias que poida ter o cliente coas partes de dito subsistema, facer máis doado de utilizar un subsistema complexo, etc.

Este patrón aplícase na definición do servizo, no cal se crean unhas interfaces contra as que ataca o controlador e as implementacións de ditas interfaces.

7.1.4 Inversion of Control (IoC)

Inversion of Control (IoC) [24] ou inversión de control é un patrón de deseño cuxa finalidade é eliminar do código as dependencias explícitas dos compoñentes, minimizando así o acoplamento.

En entornas que non utilizan a inversión de control, os propios compoñentes son os encargados de localizar a outros compoñentes dos que dependen. Mediante a inversión de control, en lugar de ser o propio compoñente o que, mediante unha infraestrutura en tempo de execución, localiza a estes outros compoñentes, é a propia infraestrutura a que se encarga de dita chamada.

A implementación do patrón *Iversion of Control* foi realizada mediante o *Framework* de *Spring*.

7.2 Arquitectura do sistema

A arquitectura global do sistema segue o patrón de arquitectura MCV, como se indica no apartado 4.1.

A continuación, explícase detalladamente cada unha das tres partes (Modelo, Vista e Controllador) que forman parte da arquitectura MVC, como se mostra na figura 4.1. Mediante estas tres partes ben diferenciadas, conseguimos que se agrupen unha serie de fucnionalidades en cada parte, co obxectivo de conseguir una aplicación modular e sinxela de manter.

7.2.1 Modelo

A capa do modelo encapsula toda a lóxica de negocio e acceso a base de datos. Dentro desta capa ocorre a creación, almacenamento, modificación e eliminación dos datos da aplicación.

Entidades persistentes

Na seguinte figura móstrase un diagrama UML de clases das entidades persistentes da aplicación.

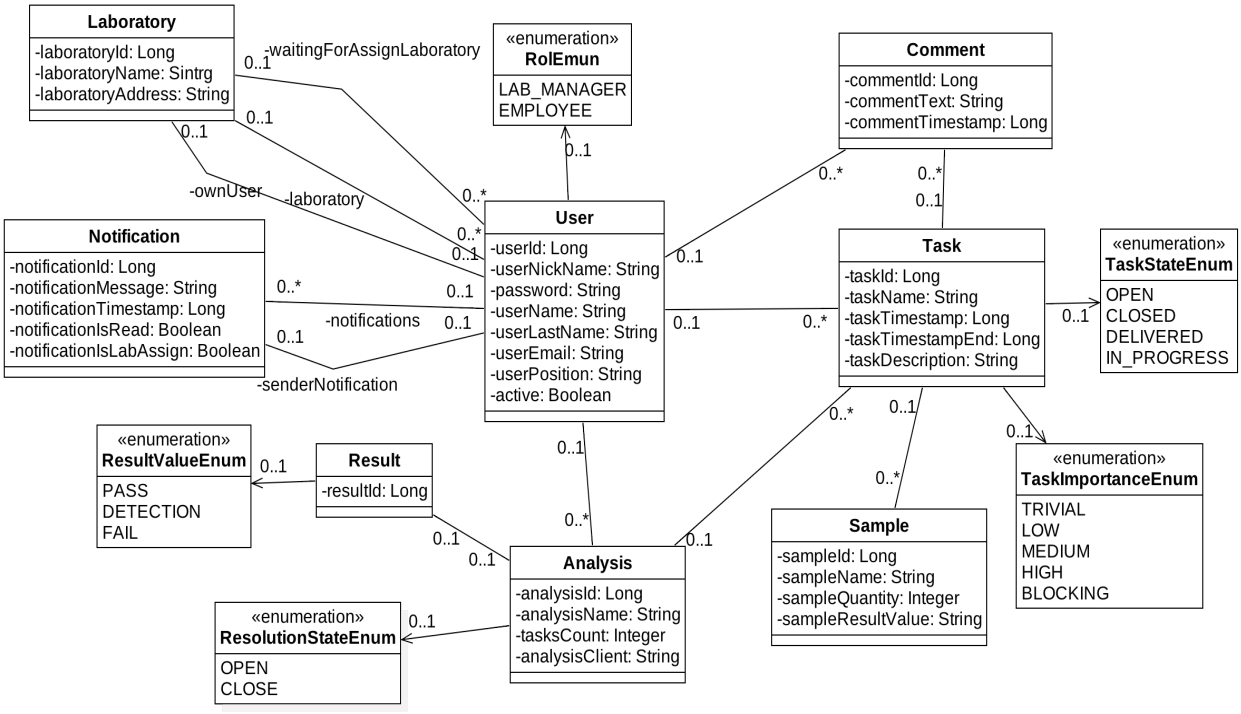


Figura 7.1: Diagrama de entidades

Detállanse brevemente cada unha das entidades mostradas:

- **User**: Datos do usuario existente no sistema.
- **Laboratory**: Datos do laboratorio existentes no sistema que pode ter un usuario.
- **Analysis**: Datos dunha análise creada por un usuario.
- **Task**: Datos dunha tarefa que pertence a unha análise.
- **Sample**: Datos das mostras que contén unha tarefa.
- **Notification**: As diferentes notificacións que se envían entre usuarios.

- **RolEnum**: Tipo enumerado que representa os posibles roles que pode ter un usuario.
- **ResolutionStateEnum**: Tipo enumerado que representa os posibles estados dun resultado.
- **Result**: Datos do resultado que poder presentar unha análise.
- **TaskStateEnum**: Tipo enumerado que representa o estado dunha tarefa.
- **TaskImportanceEnum**: Tipo enumerado que representa a importancia dunha tarefa.
- **Comment**: Datos sobre os comentarios que realiza un usuario sobre unha tarefa.

Capa de acceso a datos

O paso seguinte, a partir das clases persistentes da aplicación, é o deseño realizado na capa de acceso a datos. Esta capa fai uso do patrón *Facade*, resumido na sección 7.1.2.

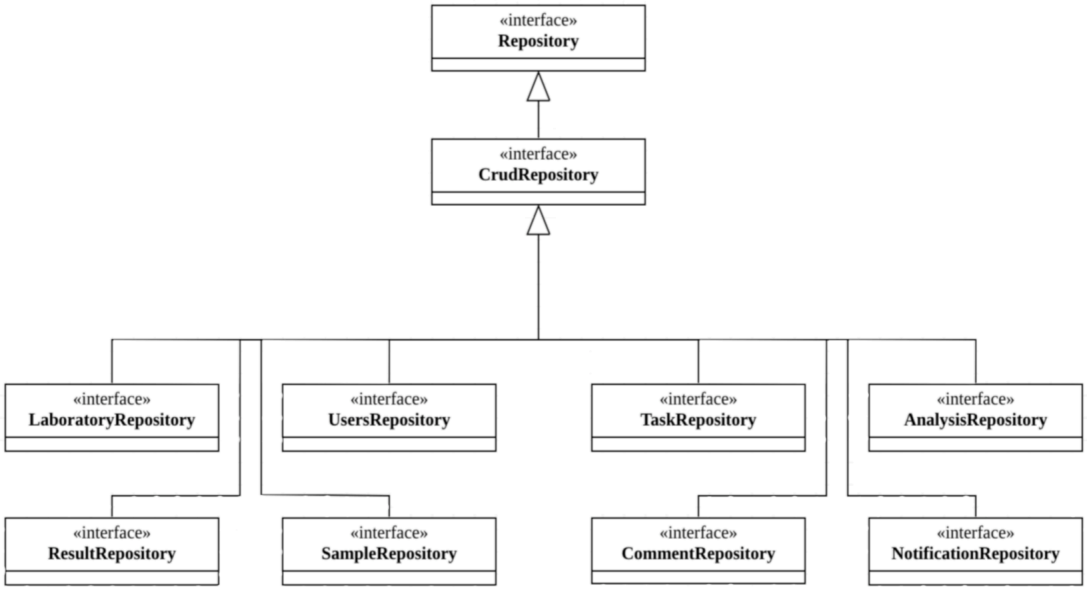


Figura 7.2: Diagrama de repositorios

Como se observa na figura 7.2, defínense todos os repositorios utilizados pola aplicación. Os repositorios propios, como poden ser UsersRepository ou SampleRepository, estenden dunha clase xenérica nomeada CrudRepository, que, á súa vez, estenden de Repository. Estas dúas clases xenéricas son de tipo ***<T, ID>***, na que o T é a entidade persistente e ID é o tipo de dato que identifica dita clase persistente. Grazas a estender de ditas clases, os repositorios propios permiten realizar as operacións **CRUD** *(CREATE, READ, UPDATE, DELETE)* sobre as entidades persistentes.

A continuación, móstrase o detalle das funcionalidades dos distintos repositorios:

UsersRepository. Este repositorio contén os métodos de acceso a datos da entidade de usuarios. Ademais dos métodos básicos *CRUD*, defínense unha serie de procuras personalizadas.

«interface» **UsersRespository**
+findByUserName(userName: String): List<User>
+findByUserEmail(userEmail: String): User
+findUserByRol(rol: RolEnum): List<User>
+findRoles(userId: long): List<RolEnum>

Táboa 7.1: UsersRepository

LaboratoryRepository. Este repositorio contén os métodos de acceso a datos da entidade de laboratorio. Ademais dos métodos básicos *CRUD*, defínese unha procura por nome de laboratorio.

«interface» **LaboratoryRepository**
+findByLaboratoryName(laboratoryName: String): List<Laboratory>

Táboa 7.2: LaboratoryRepository

AnalysisRepository. Este repositorio contén os métodos de acceso a datos da entidade análise. Ademais dos métodos básicos *CRUD*, defínense procuras por distintos criterios.

«interface» **AnalysisRepository**
+findByAnalysisName(analysisName: String): List<Analysis>
+findByUser(userId: Long): List<Analysis>
+findByUserAndState(userId: Long, state: ResolutionStateEnum): List<Analysis>

Táboa 7.3: AnalysisRepository

TaskRepository. Este repositorio contén os métodos de acceso a datos da entidade de tarefas. Ademais dos métodos básicos *CRUD*, defínense procuras por distintos criterios.

«interface» **TaskRepository**
+findByTaskName(taskName: String): List<Task> +findByTaskState(taskState: TaskStateEnum): List<Task> +findByTaskImportance(taskImportance: TaskImportanceEnum): List<Task> +findTasksByAnalysis(analysisId: Long): List<Task> +findOpenTasksByUser(userId: Long): List<Task>

Táboa 7.4: TaskRepository

SampleRepository. Este repositorio contén os métodos de acceso a datos da entidade de mostras. Ademais dos métodos básicos *CRUD*, defínese unha procura de mostras por tarefas.

«interface» **SampleRepository**
+findSamplesByTask(taskId: Long): List<Sample>

Táboa 7.5: SampleRepository

NotificationRepository. Este repositorio contén os métodos de acceso a datos da entidade de notificacións. Ademais dos métodos básicos *CRUD*, defínese unha procura de todas as notificacións dun usuario.

«interface» **NotificationRepository**
+findAllByUser(userId: Long): List<Notification>

Táboa 7.6: NotificationRepository

CommentRepository. Este repositorio contén os métodos de acceso a datos da entidade de comentarios. Non se define ningún método personalizado, só os métodos básicos *CRUD*.

«interface» **CommentRepository**

Táboa 7.7: CommentRepository

ResultRepository. Este repositorio contén os métodos de acceso a datos da entidade de comentarios. Non se define ningún método personalizado, só os métodos básicos *CRUD*.

«interface» **ResultRepository**

Táboa 7.8: ResultRepository

Capa de servizo

A capa de servizo é a encargada de implementar a lóxica de negocio e expoñela en forma de funcións, que serán chamadas por outras capas superiores. Isto realízase empregando os repositorios vistos na capa de acceso a datos do punto anterior. Os diferentes servizos están divididos de forma que cada servizo afecta a un grupo de operacións sobre a mesma entidade, é dicir, servizo de usuarios, servizo de laboratorios, etc. A continuación, detállanse en profundidade cada un destes servizos.

UserService. Este servizo contén unha serie de métodos, que se poden observar na fachada mostrada a continuación. As operacións que se poden levar a cabo con este servizo son as de engadir usuario, buscar usuarios por diferentes criterios (nome, correo electrónico, etc.), identificarse na aplicación ou actualizar usuarios, entre outras.

«interface» **UserService**
+addUser(nickName: String, password: String, rol: RolEnum, userName: String, userLastName: String, userEmail: String, userPosition: String): User +addUser(user: User): User +findUserByName(userName: String): List<User> +findUserByEmail(email: String): User +findUserById(id: Long): User +findUserByRol(rol: RolEnum): List<User> +login(email: String, clearPassword: String): User +changePassword(id: Long, oldPassword: String, newPassword: String): User +updateUser(id: Long, userNickName: String, userName: String, userLastName: String, userPosition: String, rol: RolEnum): User +deleteUser(idToDelete: Long, authId: Long): void +deleteAndAssignUser(userToDelete: User, userToAssign: User, laboratory: Laboratory): void

Táboa 7.9: UserService

LaboratoryService. O servizo de laboratorios contén as principais operacións para xestionar as entidades de laboratorios. As accións que se poden realizar son as de crear novos laboratorios, buscar laboratorios e as relacionadas coa asignación de laboratorios á usuarios.

Dentro deste grupo de operacións de asignación existen métodos para asignar, enviar correos electrónicos solicitando dita asignación e a posibilidade de marcar un laboratorio como en espera de asignación para un usuario. A interface deste servizo móstrase a continuación.

«interface» **LaboratoryService**
+addLaboratory(laboratoryName: String, laboratoryAddress: String, user: User): Laboratory
+updateLaboratory(laboratoryId: Long, laboratoryName: String, laboratoryAddress: String): Laboratory
+findByLaboratoryName(laboratoryName: String): List<Laboratory>
+assignLaboratory(laboratoryId: Long, user: User): Laboratory
+findLaboratoryById(id: Long): Laboratory
+sendAssignMail(to: String, message: String, url: String, id: Long): void
+waitingAssignLaboratory(laboratoryId: Long, user: User): Laboratory
+waitingCancelLaboratory(laboratoryId: Long, user: User): void

Táboa 7.10: LaboratoryService

AnalysisService. Os métodos definidos no servizo de análises serven para as operacións a realizar sobre as análises que os usuarios poden crear. Ditas operacións agrúpanse en creación de análises, actualizar datos, procuras por diferentes criterios, como por exemplo, por usuario, por identificador ou por estados, entre outros. Á súa vez, mediante o servizo, declárase a posibilidade de pechar e reabrir análises. A continuación, móstrase a interface do servizo.

«interface» **AnalysisService**
+createAnalysis(analysisName: String, admin: User client: String): Analysis
+findAnalysisByName(analysisName: String): List<Analysis>
+findAnalysisByUser(userId: Long): List<Analysis>
+findAnalysisByUserAndState(userId: Long, state: ResolutionStateEnum): List<Analysis>
+findAnalysisById(analysisId: Long): Analysis
+closeAnalysis(analysis: Analysis): Analysis
+reopenAnalysis(analysis: Analysis): Analysis
+changeName(analysis: Analysis, newName: String): Analysis
+deleteAnalysis(analysisId: Long): void

Táboa 7.11: AnalysisService

TaskService. O servizo de tarefas presenta similitudes co servizo de análises, xa que ditas tarefas dependen dunha análise. As operacións principais do servizo de tarefas son a creación das mesmas, procuras con diferentes criterios como o nome, importancia, entre outros, edición

de tarefas, cambios de estado e a selección dunha data de finalización coa que se creará un evento no calendario de Google, como se explicará máis adiante. A continuación, móstrase a interface do servizo.

«interface» **TaskService**
+createTask(taskName: String, taskTimestamp: Long, taskDescription: String, taskState: TaskStateEnum, taskImportance: TaskImportanceEnum, owner: User, analysis: Analysis): Task
+findByTaskName(taskName: String): List<Task>
+findByTaskState(taskState: TaskStateEnum): List<Task>
+findByTaskImportance(taskImportance: TaskImportanceEnum): List<Task>
+findAnalysisTask(analysisId: Long): List<Task>
+findTaskById(taskId: Long): Task
+addEndDate(taskId: Long, date: Date): Task
+changeTaskState(taskId: Long, newState: TaskStateEnum): Task
+editTask(taskId: Long, taskName: String, taskDescription: String): Task
+deleteTask(taskId: Long): void
+findOpenTasksByUser(userId: Long): List<Task>

Táboa 7.12: TaskService

SampleService. Este servizo mostra maior sinxeleza ca os anteriores. As operación son a creación de mostras para unha tarefa en concreto, a procura das mostras que contén unha tarefa e a posibilidade de engadir resultados a unha mostra. A continuación, móstrase a interface para o servizo de mostras.

«interface» **SampleService**
+createSample(sampleName: String, sampleQuantity: Integer, task: Task): Sample
+findSamplesByTask(taskId: Long): List<Sample>
+addSampleResult(taskId: Long, sampleId: Long, sampleResult: String): Sample
+deleteSample(sampleId: Long): void
+editSample(sampleName: String, sampleQuantity: Integer, sampleId: Long): Sample

Táboa 7.13: SampleService

CommentService. O servizo de comentarios manexa as operación sobre os comentarios que pode presentar unha tarefa. Ditas operacións son a creación de comentarios, a procura por diferentes criterios e a posibilidade de eliminar un comentario polo mesmo usuario que creou dito comentario. Os métodos están presentes na seguinte táboa da interface do servizo.

«interface» **CommentService**
+createComment(commentText: String, commentTimestamp: Long, user: User, task: Task): Comment +findCommentsByTask(taskId: Long): List<Comment> +updateComment(commentId: Long, commentText: String): Sample +findCommentById(commentId: Long): Comment +deleteComment(comment: Comment, user: User): void

Táboa 7.14: CommentService

NotificationService. Este servizo xestiona as notificacións que se envían entre usuarios. Contén os métodos para a creación de ditas notificacións, a procura das notificacións do usuario e a opción de marcar unha notificación como lida para que se lle mostre ó usuario as que xa leu nalgún momento. A continuación, móstrase a interface de dito servizo.

«interface» **NotificationService**
+createNotification(message: String, timestamp: Long, receiverUser: User, notificationIsLabAssign: Boolean, senderUser: User): Notification +findAllNotificationsFromUser(user: User): List<Notification> +findNotificationById(id: Long): Notification +markNotificationAsRead(id: Long): void

Táboa 7.15: NotificationService

ResultService. Este servizo contén os métodos para a xestión de resultados de cada análise. Permite engadir un resultado, buscar un determinado resultado dunha análise e a posible actualización de dito resultado. Os métodos móstranse na táboa do servizo a continuación.

«interface» **ResultService**
+addResult(resultValue: ResultValueEnum, analysis: Analysis): Result +findResultByAnalysis(analysisId: long): Result +updateResult(resultId: long, resultValue: ResultValueEnum): Result

Táboa 7.16: ResultService

7.2.2 Controlador

O controlador é a capa intermedia da arquitectura que actúa como ”ponte” entre a vista e o modelo. A súa finalidade é a de aceptar as petición que chegan dende a vista, xestionalas e

comunicarse coa parte do modelo correspondente a esa petición. Unha vez o modelo realiza as súas operación, o controlador devólvelle ditos datos á vista para mostralos ó usuario.

Para esta aplicación, desenvolveuse un controlador creando un servizo *REST* con diferentes *URLs* para cada funcionalidade específica que se mostran no apéndice A. Emprégase o Framework de Spring, que se encarga de recibir as petición HTTP provenientes da vista, decide que método Java da capa de servizos é necesario empregar e reponde coa saída de dito método. A continuación, detállanse os controladores que existen na aplicación:

- **UserController**. O controlador de usuarios agrupa todas as funcionalidades que teñen relación coas actividades dos usuarios. A funcionalidade do controlador engloba as seguintes accións:
  - Autenticación de usuarios e a posible creación en caso de un rexistro novo.
  - Saír da aplicación.
  - Actualizar os datos perfil de usuario.
  - etc.
- **LaboratoryController**. A función do controlador de laboratorios é interceptar as petición sobre as accións sobre os mesmos. A funcionalidade é a seguinte:
  - Visualizar un laboratorio.
  - Crear un laboratorio.
  - Permitir solicitar a asignación a un laboratorio, así como poder aceptar dita solicitude ou rexeitala.
  - Manexar o formulario de procuras.
  - etc.
- **AnalysisController**. Mediante o controlador de análises, pódese realizar todas as accións sobre as análises, incluíndo a exportación de PDF. A principais funcionalidades son:
  - Visualizar análises.
  - Crear unha nova análise.
  - Cambiar o estado.
  - Editar a análise.
  - Exportar os datos da análise en formato PDF.
  - etc.

- **TaskController**. O controlador de tarefas engloba tanto a funcionalidade sobre tarefas como sobre mostras. Algunhas da funcións son:
  - Creación e visualización de tarefas.
  - Edición de tarefa.
  - Cambio de estado sobre as tarefas.
  - Inserción de mostras á tarefa.
  - Engadir resultados ás mostras.
  - Comentar sobre unha tarefa.
  - etc.
- **NotificationController**. A función deste controlador é a de xestionar as notificacións, dende a creación ata a visualización das mesmas.
- **HomeController**. O controlador do *HOME* é un controlador especial, cuxa función é a de comprobar se un usuario que accede á aplicación o fai por primeira vez ou non. Dependendo diso, mostra un formulario para completar unha serie de datos adicionais, se é a primeira vez que o usuario accede, ou a páxina de inicio no caso contrario.
- **GoogleCalendarController**. Este controlador xestiona a comunicación coa API Calendar de Google, posibilitando a creación de eventos a través dunha tarefa. Recolle na petición a data de finalización dunha tarefa, que selecciona un usuario, e envía dita data á API para proceder a crear o evento.

7.2.3 Vista

A vista é a última capa da arquitectura. É a capa que interactúa co usuario, mostrándolle información do sistema e enviando as peticións do usuario ó controlador.

Debido a que aplicación é web, a capa da vista vaise executar nun navegador web, polo que a vista estará composta por páxinas *HTML*, con *CSS* para os estilos e *JavaScript* para acadar dinamismo dentro da páxina estática. Para a implementación desta capa, emprégase o framework *Thymeleaf*, debido a súa eficiencia á hora de implementarse nos arquivos *HTML*.

A ampla usabilidade de Thymeleaf ven dada polo concepto de *Plantillas Naturais*. Mediante este procedemento conséguese inxectar a lóxica propia do framework no arquivo *HTML*. A forma de inxectar os compoñente propios de *Thymeleaf* é mediante o texto **th:** seguido da etiqueta que se desexe. Por exemplo un texto fórmase como *th:text*.

A forma na que *Thymeleaf* accede a datos provenientes do código Java é mediante caracteres especiais, como o **$** ou **#** entre outros. A continuación, móstranse exemplos dalgunhas das instrucións máis empregadas:

- Para acceder ó nome dunha tarefa: ***th:utext="${task.taskName}"***.
- Para acceder a datos provenientes do ficheiro de properties: ***th:utext="#{properties.alert}"***.
- Thymeleaf dispón tamén de etiquetas "lóxicas" como if: ***th:if="#{analysis.size() > 0}"***.
- A forma na que Thymeleaf pode comprobar se un usuario está autenticado na aplicación: ***sec:authorize="isAuthenticated()"***.

A continuación, móstrase a estrutura dun proxecto Spring Boot con Thymeleaf, cos directorios principais. O directorio principal no que se sitúan os ficheiros necesarios para a vista é **src/main/resources**.

- **static**. Directorio onde se sitúan os elementos estáticos da páxina, como poden ser as fontes, scripts de *JavaScript* e estilos *CSS*.
- **templates**. Directorio onde se atopan os propios ficheiros *HTML* coas anotación de Thymeleaf.

7.3 Integración coa API Calendar de Google

Neste proxecto decidiuse implementar a integración coa API Calendar de Google. Esta API permite que un usuario autenticado mediante Google, poida xestionar eventos no seu calendario persoal, en forma de recordatorios.

A pesar de que a API permite diversas accións, como crear, eliminar ou modificar eventos, decidiuse centrala só na creación. Esta decisión ven tomada pola necesidade de controlar as datas nas que finaliza unha tarefa. Desta maneira, cando o usuario selecciona unha data de finalización rexístrase un novo evento no calendario do usuario. Ese evento serve como recordatorio, debido a que o usuario non ten que estar pendente de todas as tarefas, xa que a API envía unha notificación e un correo electrónico cando se acerca a data de finalización de cada tarefa.

Para a propia integración da API, é necesario realizar uns pasos previos:

As fases necesarias son, nun inicio, dar de alta a aplicación no portal de *Google Developers*, agregando no dito portal información sobre *URLs* de aceso e de re-direción.

Unha vez realizado o anterior, modificouse o propio *Login* da aplicación para re-dirixir ao usuario a través do login externo.

Para a integración da API no sistema é necesario que o usuario conte con unha conta de Google.

Capítulo 8

Implementación e probas

A continuación, expóñense os detalles do proceso que se leva a cabo para a codificación, compilación e execución do proxecto.

8.1 Software requirido

Para a codificación do proxecto empréganse as seguintes ferramentas:

- **Eclipse IDE**: Versión 2019-09 R (4.13.0)
- **JDK**: Versión 1.8.0_222
- **MySQL**: Versión 8.0.19
- **Apache Maven**: versión 4.0.0
- **Spring Boot**: Versión 2.0.2 RELEASE
- **Spring Boot - Thymeleaf**: Versión 2.0.2 RELEASE
- **MySQLWorkbench**: versión 8.0.19
- **Bootstrap**: versión 4.0.0-2

8.2 Estrutura do proxecto

Na estrutura do proxecto diferéncianse dúas partes. Por un lado está o código *Java*, que se encarga da lóxica de negocio, acceso a datos, repositorios, servizos, etc. Por outro lado está a parte empregada por *Thymeleaf*, é dicir, os ficheiros *HTML*, *CSS* e *JavaScript*. Esta última parte é o que representa a interface de usuario.

8.2.1 Estrutura do proxecto Java (Modelo e Controlador)

A estrutura do proxecto está deseñada segundo un arquetipo estándar de *Maven*. No directorio raíz do proxecto encóntrase un ficheiro nomeado ”pom.xml”, no cal se encontra a configuración de *Maven*. Á súa vez, neste ficheiro, xestiónanse as dependencias con outros softwares que son necesarios para a aplicación, ademais de incluír a forma na que se compila e desprega o proxecto.

No directorio **src/main/java** encóntrase o código fonte *Java* que forma parte do modelo e do controlador. Está organizado por paquetes que conteñen diferentes clases agrupadas dependendo da funcionalidade que cumpren. A continuación, explícase cada un destes paquetes:

- **com.lab**. Neste paquete só se encontra a clase Application.java, necesaria para poder lanzar a aplicación *Spring Boot*.
- **com.lab.business**. Paquete que contén as interfaces do servicio.
- **com.lab.business.impl**. Está relacionado co paquete com.lab.business, debido a que contén as implementación das interfaces citadas no paquete anterior.
- **com.lab.business.entities**. Inclúe todas as entidades que emprega a aplicación, como poden ser tarefas, análises, usuarios, etc.
- **com.lab.business.repository**. Neste paquete encóntranse as interfaces dos repositorios empregados pola aplicación, como UsersRepository ou AnalysisRepository, entre outros.
- **com.lab.business.exceptions**. Contén as clases de excepcións que pode lanzar a aplicación nalgún momento dado.
- **com.lab.business.utils**. Paquete que contén clases de utilidade que pode necesitar a aplicación.
- **com.lab.common**. Contén a clase ConfigurationParameters e Constants. Son clases comúns a toda a aplicación.
- **com.lab.rest**. Neste paquete inclúense os controladores da aplicación.
- **com.lab.web.conf**. Paquete coas clases de configuración do contexto de Spring, como poden ser as de seguridade.
- **com.lab.web.exceptions**. Contén as excepcións que se capturan na capa web da aplicación e como manexalas.

- **com.lab.web.forms**. Este paquete contén os formularios que emprega o controlador *Java* para recibir as peticións do usuario provenientes da vista e recuperar así ditos datos para envialos ó modelo.
- **com.lab.web.secutiry.oauth2**. Contén as clases necesarias para poder iniciar sesión mediante oAuth2 de Google.

8.2.2 Estrutura da Vista Thymeleaf

No directorio **src/main/resources** encóntranse os ficheiros co código *HTML*, *CSS* e *JavaScript* que se empregan para o interface de usuario. Como se comentou anteriormente, nos ficheiros *HTML* están presente os comandos propios de Thymeleaf cos que se constrúe a vista.

A continuación, detállanse as funcións de cada carpeta:

- **static**. Contén os recursos estáticos que empregará a vista. Estes recursos son:
  - Fontes.
  - Ficheiros de estilos (*CSS).*
  - Scripts *JavaScript.*
- **templates**. Os propios ficheiros co código *HTML* que forman a interface de usuario.
- **Ficheiros .properties**. Conteñen os textos e mensaxes que se mostran ó usuario na vista. Actualmente existen dous, mensaxes en castelán e mensaxes en inglés.

8.3 Instrucións de compilación e execución

Antes de comentar os pasos necesarios para a compilación e execución do proxecto, é necesario unha serie de requisitos relacionados coa base de datos MySQL.

- Un servicio MySQL executándose na maquina local.
- Unha base de datos dentro do servicio MySQL creada co nome **tfg**. Os datos de acceso a dita base de datos son:
  - Usuario: tfguser
  - Contrasinal: tfg

En canto á propia compilación e execución, os pasos son os seguintes:

1. Abrir unha consola/terminal no directorio raíz do proxecto, neste caso **lab-manager-app**.

2. Executar o comando **mvn spring-boot:run**.

3. Abrir nun navegador web a páxina **localhost:8000** para acceder á aplicación.

8.4 Probas

Unha das fases máis importantes durante o desenvolvemento da aplicación é a realización de probas dentro dun ciclo iterativo. Mediante as probas conséguese asegurar que a implementación non contén erros e está correcta conforme ós criterios de aceptación acordados co cliente.

A continuación, móstranse os distintos tipos de probas levadas a cabo durante cada ciclo do proceso de implementación da aplicación.

8.4.1 Probas unitarias

O obxectivo das probas unitarias é o de comprobar o correcto funcionamento de elementos illados do código, como bloques ou funcións do código *Java*. Estas probas deben ser realizadas, preferiblemente, de forma illada, sen que afecten a outros compoñentes do código, e que se poidan repetir tantas veces como sexa necesario sen interferir en outras probas ou resultados do sistema xeral.

As probas unitarias realizáronse sobre os métodos definidos nos servizos de usuario, laboratorio, análise, mostras, resultados, comentarios e tarefas. A maioría das funcionalidades destes servizos son creacións, eliminacións ou procuras, polo que ditas probas unitarias centráronse máis neses aspectos, probando con diferentes parámetros e casuísticas ditos métodos.

As probas unitarias son realizados polo alumno, como rol de programador, de forma manual. A batería destas probas realízanse no directorio **src/test/java**.

8.4.2 Probas de integración

O seguinte paso despois das probas unitarias son as probas de integración. Estas probas diferéncianse das anteriores en que xa non se realizan con elementos illados do código, senón con grupos de elementos. A forma correcta de realizar estas probas é comprobando que dous ou máis elementos illados funcionan correctamente executándose xuntos, é dicir, céntranse na correcta comunicación entre compoñentes.

Estas probas son tamén realizadas polo alumno, como rol de programador, de forma manual cunha batería de probas definidas como se comentou no punto anterior.

8.4.3 Probas de sistema

As probas de sistema consisten en probar o correcto funcionamento do sistema global, é dicir, modelo, vista e controlador. A finalidade é comprobar que o conxunto da aplicación funciona sen erros na súa totalidade como un sistema final.

Estas probas realízanse mediante un navegador web *Google Chrome*, recorrendo as pantallas que forman a interface da vista, validando as distintas funcionalidades.

8.4.4 Probas de aceptación

Por último, unha vez rematado o desenvolvemento, realízanse as probas de aceptación. Estas probas lévaas a cabo o cliente, aínda que neste caso realízanas o alumno e o director do proxecto.

O obxectivo destas probas de aceptación é comprobar se o sistema final cumpre cos requisitos e funcionalidades demandados inicialmente. Nese caso as probas finalizarán con resultado correcto.

Capítulo 9

Conclusións

9.1 Conclusións finais

O sistema final cumpre cos requirimentos previamente marcados. A aplicación web resultante permite a xestión de análises, tarefas e mostras dun laboratorio, empregando unha interface gráfica clara, amigable e intuitiva.

Durante o desenvolvemento xurdiron algunhas dificultades, como por exemplo a integración coas *APIs* Google. Esta integración resultou máis complexa do que nun principio se estimara, sobre todo a realización do *login* de Google. A pesar disto, o desenvolvemento foi abordado e executado correctamente.

A interface web proporciona aos usuarios a información que se almacena nas entidades persistentes. Para o deseño empregáronse modelos *HTML* con atributos de *Thymeleaf*, dándolle estilos con *CSS* e funcionalidade mediante *JavaScript*, así como *plugins* de *jQuery* como *DataTables*.

Á súa vez, deseñouse unha *API REST* seguindo as guías de calidade do estándar. Esta *API* é consumida pola vista para recuperar os datos persistentes. Realizouse empregando as tecnoloxías de *Java* e *Spring Boot*, que permiten minimizar, notablemente, o tempo de desenvolvemento e posta en marcha do proxecto. O modelo contra o que ataca a *API REST* deseñouse e implementouse nunha base de datos *MySQL* na que encaixa a funcionalidade requirida.

En canto a utilización da *API* externa de *Calendar* de Google, implementouse a creación de eventos de forma sinxela para o seguimento de tarefas.

Cabe destacar tamén a internacionalización (*i18n*) da aplicación, coa cal se traducen todos os textos e etiquetas aos idiomas castelán e inglés.

No tocante ó desenvolvemento, a utilización de tecnoloxías modernas e próximas ó mundo

laboral, posibilitou a adquisición de novos conceptos, coñecementos e experiencia.

O obxectivo deste proxecto foi desenvolver unha aplicación sendo o máis rigoroso e o máis próximo ó mundo real posible, ademais de aprender do proceso levado a cabo.

9.2 Futuras liñas de traballo

A continuación, móstranse posibles liñas de traballo que se poden implementar nun futuro, mellorando así o produto:

- Melloras visuais na interface.
- Implementación dun chat interno evitando o uso de correos electrónicos para a comunicación entre usuarios.
- Introducir gráficas estatísticas que mostren, por exemplo, tarefas por usuario ou resultados positivos contra negativos.
- Ofrecer outras alternativas de idiomas, coma o galego.

Apéndices

Apéndice A

API REST

/login	
Descrición	Recupera o formulario para identificarse no sistema.
Método	GET.
Parámetros	N/A.
Formulario	N/A.

Táboa A.1: Recuperar páxina de login

/login	
Descrición	Permite ó usuario identificarse no sistema.
Método	POST.
Parámetros	N/A.
Formulario	LoginForm.

Táboa A.2: Login de usuario

/logout	
Descrición	Pecha a sesión do usuario do sistema.
Método	GET.
Parámetros	N/A.
Formulario	N/A.

Táboa A.3: Logout do usuario

/profile/delete	
Descrición	Elimina o perfil do usuario do sistema.
Método	POST.
Parámetros	N/A.
Formulario	UserProfileForm.

Táboa A.4: Eliminar usuario

/laboratory/lab	
Descrición	Recupera o laboratorio do usuario.
Método	GET.
Parámetros	N/A.
Formulario	N/A.

Táboa A.5: Recuperar laboratorio de usuario

/laboratory/lab/{id}/edit	
Descrición	Recupera o formulario para editar o laboratorio.
Método	GET.
Parámetros	id: identificador do laboratorio.
Formulario	N/A.

Táboa A.6: Recuperar formulario para editar laboratorio

/laboratory/lab/{id}/edit	
Descrición	Edita a información do laboratorio.
Método	POST.
Parámetros	id: identificador do laboratorio.
Formulario	EditLaboratoryForm.

Táboa A.7: Editar a información do laboratorio

/laboratory/add	
Descrición	Recupera o formulario para engadir un laboratorio.
Método	GET.
Parámetros	N/A.
Formulario	N/A.

Táboa A.8: Recuperar formulario para engadir laboratorio

/laboratory/add	
Descrición	Engade un novo laboratorio.
Método	POST.
Parámetros	N/A.
Formulario	AddLaboratoryForm.

Táboa A.9: Engadir laboratorio

/laboratory/search	
Descrición	Recupera o formulario de procura dun laboratorio.
Método	GET.
Parámetros	N/A.
Formulario	N/A.

Táboa A.10: Recuperar formulario de procura de laboratorio

/laboratory/search	
Descrición	Procura un laboratorio.
Método	POST.
Parámetros	N/A.
Formulario	SearchLaboratoryForm.

Táboa A.11: Procurar un laboratorio

/laboratory/assign/{id}	
Descrición	Solicita a asignación a un laboratorio.
Método	POST.
Parámetros	id: identificador do laboratorio.
Formulario	N/A.

Táboa A.12: Asignación a un laboratorio

/laboratory/assign/cancel/{id}	
Descrición	Cancela a solicitude de asignación a un laboratorio.
Método	POST.
Parámetros	id: identificador do laboratorio.
Formulario	N/A.

Táboa A.13: Cancelar a solicitude de asignación a un laboratorio

/profile	
Descrición	Recupera a información de perfil dun usuario.
Método	GET.
Parámetros	N/A.
Formulario	N/A.

Táboa A.14: Recuperar información de perfil

/profile	
Descrición	Actualiza a información de perfil dun usuario.
Método	POST.
Parámetros	N/A.
Formulario	UserProfileForm.

Táboa A.15: Actualizar información de perfil

/analysis	
Descrición	Recupera as análises dun usuario.
Método	GET.
Parámetros	N/A.
Formulario	N/A.

Táboa A.16: Recuperar análises de usuario

/analysis/{id}	
Descrición	Recupera a información dunha análise.
Método	GET.
Parámetros	id: identificador da análise.
Formulario	N/A.

Táboa A.17: Recuperar unha análise

/analysis/{id}/add/result	
Descrición	Engade un resultado a unha análise.
Método	POST.
Parámetros	id: identificador da análise.
Formulario	AddResultForm.

Táboa A.18: Engadir resultado a unha análise

/analysis/{id}/tasks	
Descrición	Recupera as tarefas dunha análise.
Método	GET.
Parámetros	N/A.
Formulario	N/A.

Táboa A.19: Recuperar tarefas dunha análise

/analysis/create	
Descrición	Recupera o formulario para a creación dunha nova análise.
Método	GET.
Parámetros	N/A.
Formulario	N/A.

Táboa A.20: Recuperar formulario de creación de análise

/analysis/create	
Descrición	Crea unha nova análise.
Método	POST.
Parámetros	N/A.
Formulario	AddAnalysisForm.

Táboa A.21: Creación de análise

/analysis/{id}/close	
Descrición	Cambia o estado a pechado dunha análise.
Método	POST.
Parámetros	id: identificador da análise.
Formulario	N/A.

Táboa A.22: Pechar análise

/analysis/{id}/reopen	
Descrición	Cambia o estado a aberto dunha análise.
Método	POST.
Parámetros	id: identificador da análise.
Formulario	N/A.

Táboa A.23: Reabrir análise

/analysis/{id}/delete	
Descrición	Elimina unha análise.
Método	POST.
Parámetros	id: identificador da análise.
Formulario	N/A.

Táboa A.24: Eliminar análise

/notifications	
Descrición	Recupera as notificacións dun usuario.
Método	GET.
Parámetros	N/A.
Formulario	N/A.

Táboa A.25: Recuperar notificacións

/notifications/{id}	
Descrición	Recupera unha notificación dun usuario.
Método	GET.
Parámetros	N/A.
Formulario	N/A.

Táboa A.26: Recuperar unha notificación

/laboratory/{labId}/assign/accept/{userId}	
Descrición	Acepta a solicitude de asignación a un laboratorio dun usuario.
Método	POST.
Parámetros	labId: identificador do laboratorio. userId: identificador do usuario pendente da asignación.
Formulario	N/A.

Táboa A.27: Aceptar unha solicitude de asignación

/laboratory/{labId}/assign/decline/{userId}	
Descrición	Rexeita a solicitude de asignación a un laboratorio dun usuario.
Método	POST.
Parámetros	labId: identificador do laboratorio. userId: identificador do usuario pendente da asignación.
Formulario	N/A

Táboa A.28: Rexeitar unha solicitude de asignación

/task/create/analysis/{id}	
Descrición	Recupera o formulario de creación de tarefas.
Método	GET.
Parámetros	id: identificador da análise.
Formulario	N/A.

Táboa A.29: Recuperar formulario de creación de tarefas

/task/create/analysis/{id}	
Descrición	Crea unha nova tarefa.
Método	POST.
Parámetros	id: identificador da análise.
Formulario	AddTaskForm.

Táboa A.30: Creación de tarefas

/task/{id}	
Descrición	Recupera a información de unha tarefa.
Método	GET.
Parámetros	id: identificador da tarefa.
Formulario	N/A.

Táboa A.31: Recuperación dunha tarefa

task/{id}/add/comment	
Descrición	Engade un comentario a unha tarefa.
Método	POST.
Parámetros	id: identificador da tarefa.
Formulario	AddCommentForm.

Táboa A.32: Engadir comentario a unha tarefa

task/{id}/delete/comment	
Descrición	Elimina un comentario dunha tarefa.
Método	POST.
Parámetros	id: identificador da tarefa.
Formulario	DeleteCommentForm.

Táboa A.33: Eleminar comentario dunha tarefa

/analysis/{id}/changename	
Descrición	Modifica o nome dunha análise.
Método	POST.
Parámetros	id: identificador da análise.
Formulario	ChangeAnalysisNameForm.

Táboa A.34: Modificar o nome dunha análise

/task/{id}/enddate	
Descrición	Recupera o formulario para engadir unha data de finalización a unha tarefa.
Método	GET.
Parámetros	id: identificador da tarefa.
Formulario	N/A.

Táboa A.35: Recuperar formulario de data de finalización

/task/{id}/enddate	
Descrición	Engade unha data de finalización a unha tarefa.
Método	POST.
Parámetros	id: identificador da tarefa.
Formulario	AddTaskEndateForm.

Táboa A.36: Engadir data de finalización

/newuser/update	
Descrición	Recupera o formulario de rexistro por primeira vez.
Método	GET.
Parámetros	N/A.
Formulario	N/A.

Táboa A.37: Recuperar o formulario de novo rexistro

/newuser/update	
Descrición	Actualiza os datos dun usuario que se rexistra por primeira vez no sistema.
Método	POST.
Parámetros	N/A.
Formulario	UserNewProfileForm.

Táboa A.38: Actualizar datos de novo rexistro

/task/{id}/change/state	
Descrición	Permite cambiar o estado dunha tarefa.
Método	POST.
Parámetros	id: identificador da tarefa.
Formulario	ChangeTaskStateForm.

Táboa A.39: Cambiar estado dunha tarefa

/task/{id}/add/sample/result	
Descrición	Engade un resultado a unha mostra.
Método	POST.
Parámetros	id: identificador da tarefa.
Formulario	AddSampleResultForm.

Táboa A.40: Engadir resultado a unha mostra

/task/{id}/add/sample	
Descrición	Recupera o formulario para engadir unha a unha tarefa.
Método	GET.
Parámetros	id: identificador da tarefa.
Formulario	N/A.

Táboa A.41: Recuperar formulario para engadir unha mostra

/task/{id}/add/sample	
Descrición	Recupera o formulario para engadir unha a unha tarefa.
Método	POST.
Parámetros	id: identificador da tarefa.
Formulario	AddSampleForm.

Táboa A.42: Engadir unha mostra

/analysis/{id}/exportPdf/checks	
Descrición	Comproba que os requisitos da análise para a súa exportación son os correctos.
Método	GET.
Parámetros	id: identificador da análise.
Formulario	N/A.

Táboa A.43: Comprobación da análise para exportar

/analysis/{id}/exportPdf	
Descrición	Exporta os datos dunha análise en formato PDF.
Método	GET.
Parámetros	id: identificador da análise.
Formulario	N/A.

Táboa A.44: Exportación dunha análise en PDF

/task/{id}/edit	
Descrición	Edita a información dunha tarefa.
Método	POST.
Parámetros	id: identificador da tarefa.
Formulario	EditTaskForm.

Táboa A.45: Editar tarefa

/task/{id}/delete	
Descrición	Elimina unha tarefa.
Método	POST.
Parámetros	id: identificador da tarefa.
Formulario	N/A.

Táboa A.46: Eliminar tarefa

/task/{id}/delete/sample	
Descrición	Elimina unha mostra dunha tarefa.
Método	POST.
Parámetros	id: identificador da tarefa.
Formulario	DeleteSampleForm.

Táboa A.47: Eliminar mostra

/task/{id}/edit/sample	
Descrición	Modifica a información dunha mostra dunha tarefa.
Método	POST.
Parámetros	id: identificador da tarefa.
Formulario	EditSampleForm.

Táboa A.48: Editar mostra

/tasks/open	
Descrición	Recupera as tarefas en estado aberto de un usuario.
Método	GET.
Parámetros	N/A.
Formulario	N/A.

Táboa A.49: Recuperar tarefas abertas

/profile/delete/newOwner	
Descrición	Recupera a lista de usuario dun mesmo laboratorio.
Método	GET.
Parámetros	N/A.
Formulario	N/A.

Táboa A.50: Recuperar usuarios dun laboratorio

/profile/delete/newOwner	
Descrición	Asigna a un novo usuario a propiedade sobre o laboratorio.
Método	POST.
Parámetros	N/A.
Formulario	UserNewOwnerForm.

Táboa A.51: Asignar un novo usuario propietario

Apéndice B

Manual de usuario

A continuación, móstrase un manual de usuario detallando as principais funcionalidades da aplicación web. Para acceder a aplicación é necesario acceder dende un navegador web á *URL* http://localhost:8000/.

Cabe aclarar que para todas as interacción que realiza o usuario coa aplicación móstrase sempre unha mensaxe informativa referente a dita acción e, no caso de que se produza algún erro, informaráselle ó usuario o motivo e as posibles solucións.

B.1 Páxina de inicio

B.1.1 Páxina de inicio sin autenticación

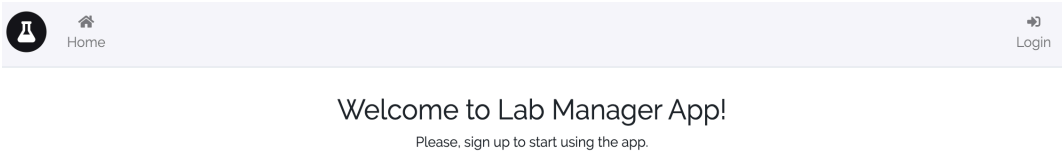


Figura B.1: Páxina de inicio sen autenticación

B.1.2 Páxina de inicio con autenticación

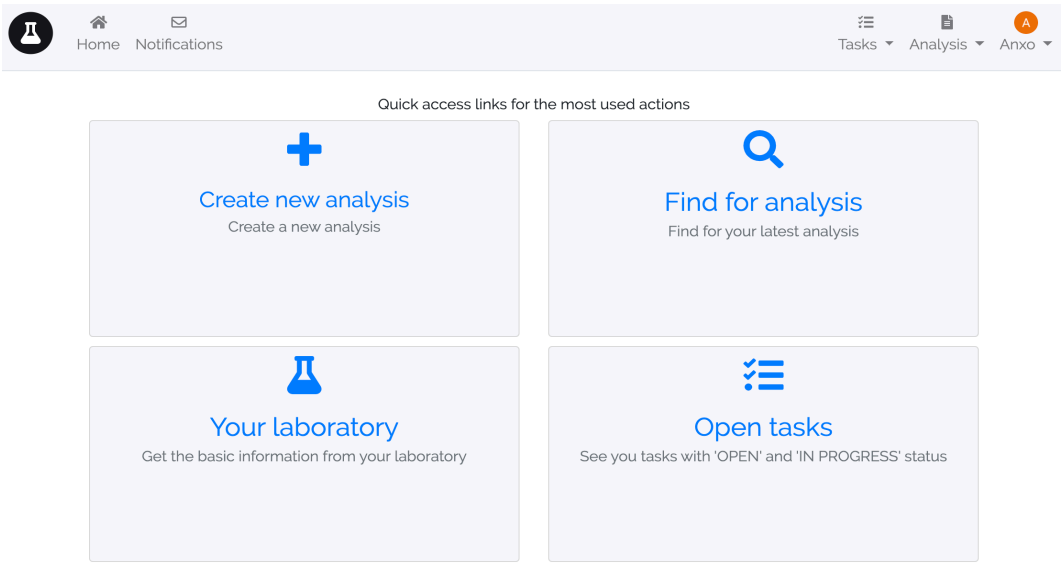


Figura B.2: Páxina de inicio con autenticación

B.2 Perfil

B.2.1 Rexistro

Na páxina de rexistro, móstrase un enlace para acceder á aplicación mediante *Google*.

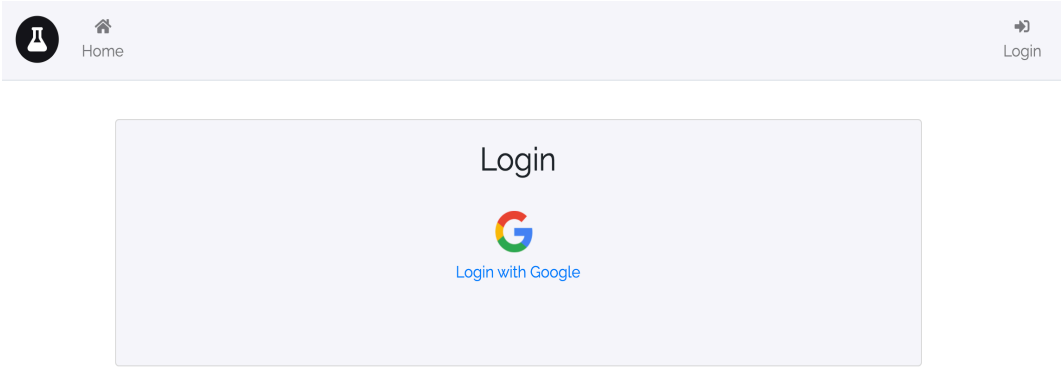


Figura B.3: Páxina de rexistro 1

O usuario preme na icona de Google para acceder ó rexistro mediante o servizo de Google.

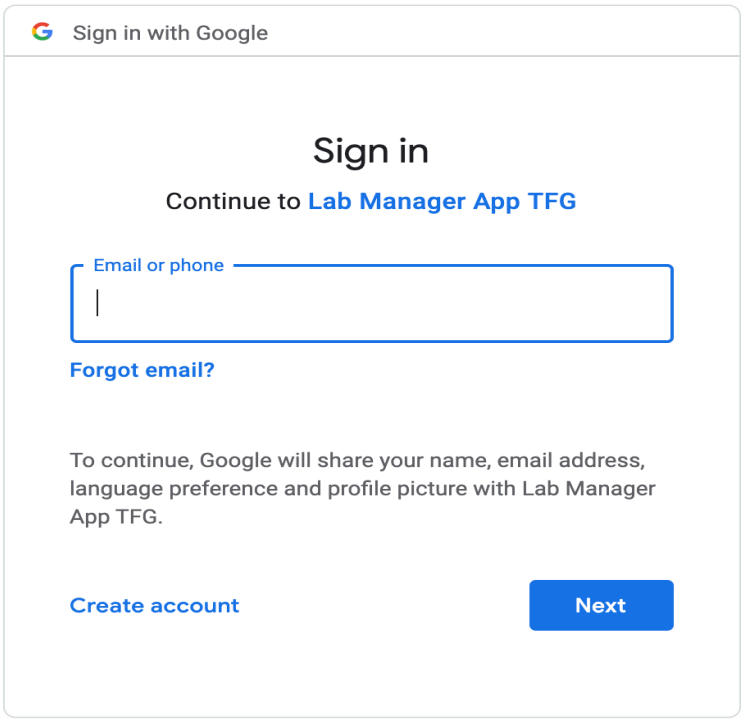


Figura B.4: Páxina de rexistro 2

Unha vez introducidos os datos, o usuario é re-dirixido a unha páxina na que se solicita completar certos datos persoais internos para a aplicación.

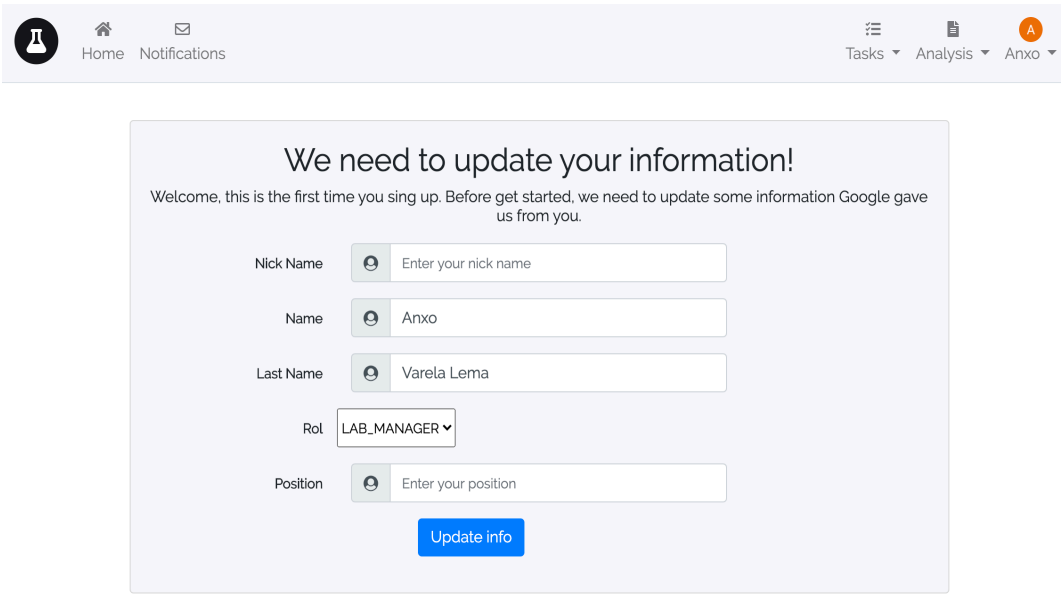


Figura B.5: Páxina de rexistro 3

O usuario cubre os campos baleiros e/ou actualiza os restantes.

B.2.2 Perfil

O usuario desprega o menú premendo no seu nome na barra do menú. Móstranse diferentes opcións. Neste caso, a indicada é ”Perfil”.

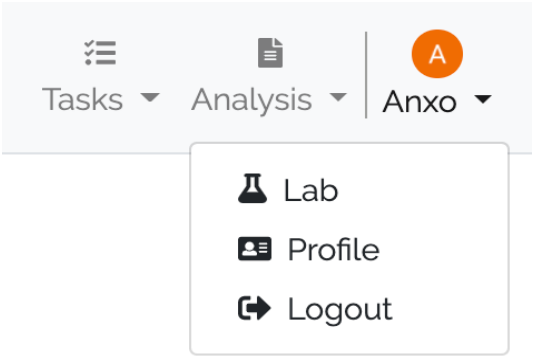


Figura B.6: Menú do usuario

Móstranse os datos con posibilidade de actualizar do usuario. Existe a posibilidade de modificar os campos que se desexen. Unha vez actualizados, o usuario preme en ”Actualizar” e os novos campos son gardados.

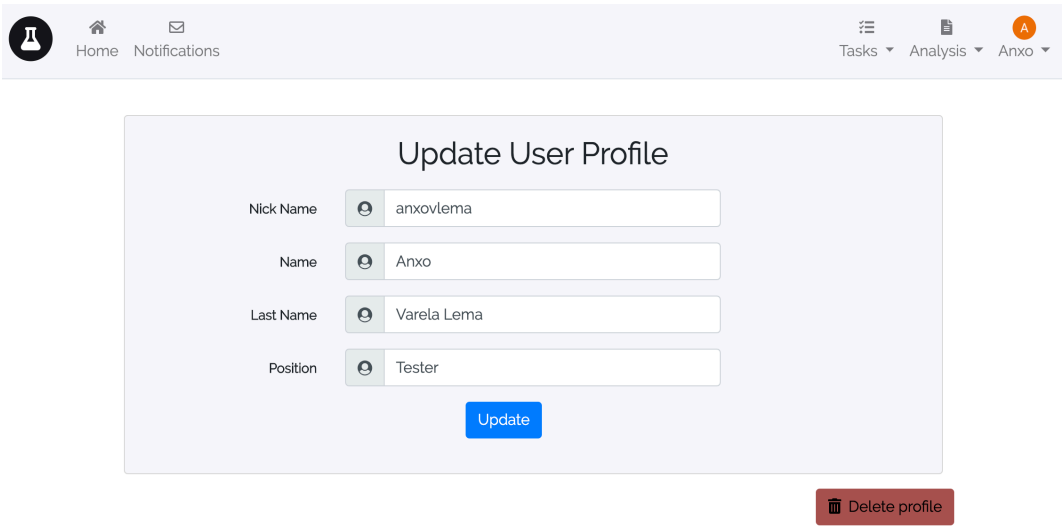


Figura B.7: Perfil do usuario

Se o usuario desexa eliminar a súa conta do sistema, preme no botón ”Eliminar perfil”, o cal fai que se mostre unha modal indicando se está seguro da acción a realizar.

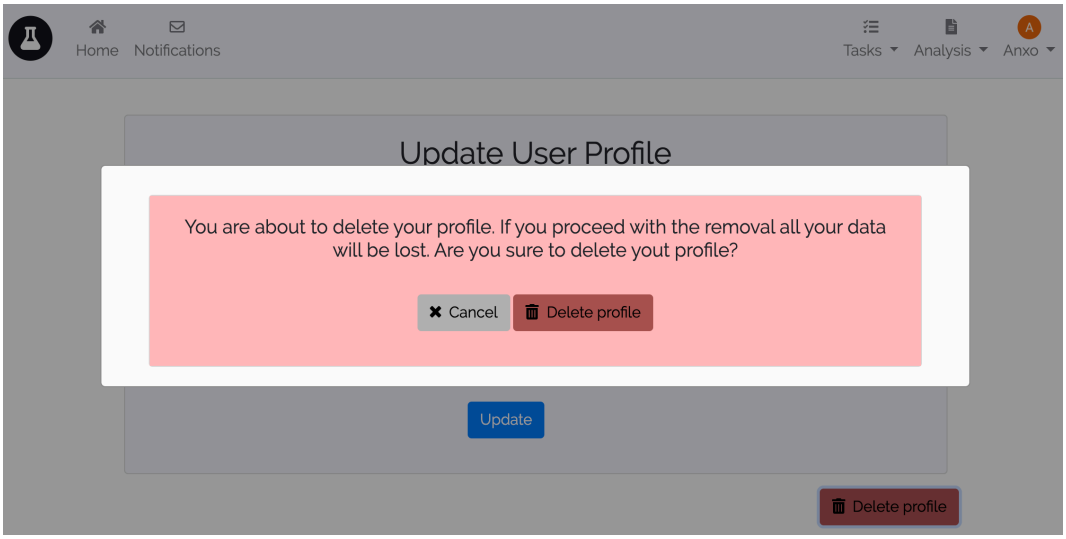


Figura B.8: Eliminar perfil do usuario

B.3 Laboratorio

B.3.1 Buscar laboratorio

O usuario desprega o menú premendo no seu nome na barra do menú. Móstranse diferentes opcións. Neste caso, a indicada é Laboratorio.

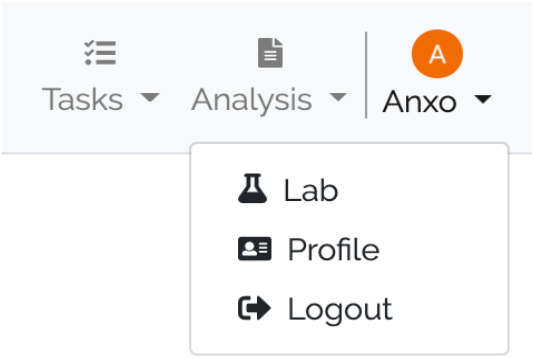


Figura B.9: Menú do usuario

Unha vez pulsada a opción de ”Lab”, o sistema mostra as opcións a realizar sobre os laboratorios, neste caso só a de buscar, debido a que o usuario rexistrado ten rol de empregado.

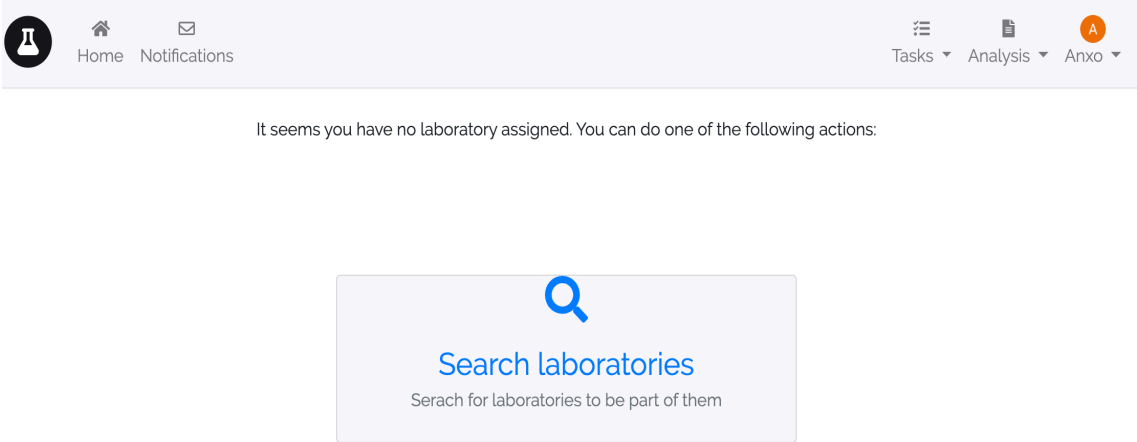


Figura B.10: Buscar laboratorio 1

Ao premer na opción de buscar laboratorios, móstrase un formulario de procura, cun campo para o nome do laboratorio.

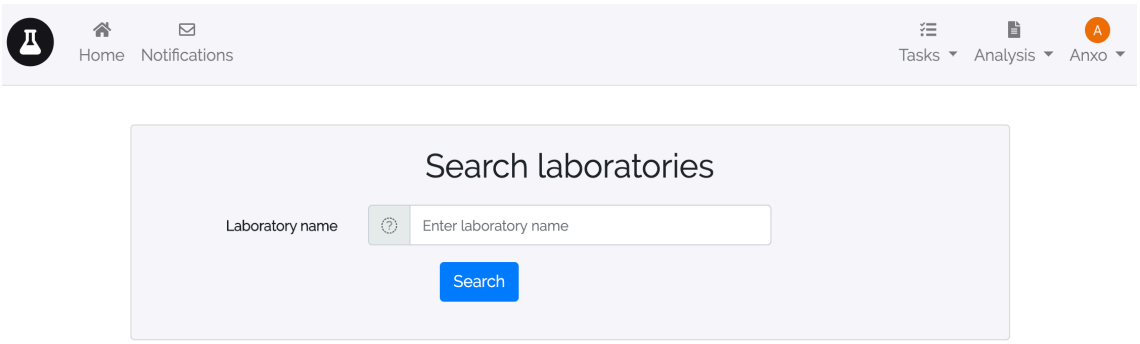


Figura B.11: Buscar laboratorio 2

Unha vez inserido o nome a buscar, móstrase unha lista cos resultados que coinciden co criterio.

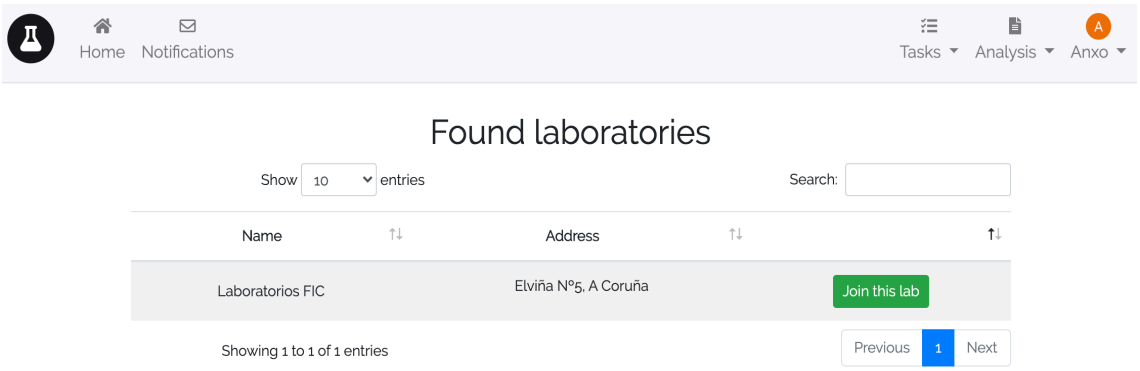


Figura B.12: Buscar laboratorio 3

Para unirse ó laboratorio desexado, o usuario premerá no botón de unirse ó laboratorio que se mostra en cada entrada da lista. A continuación, no apartado B.3.2, detállase o procedemento de asignación.

B.3.2 Unirse a un laboratorio

Unha vez seleccionado o laboratorio desexado e solicitado a incorporación do usuario a dito laboratorio, móstrase unha mensaxe indicando que o usuario debe agardar ata que o encargado do laboratorio acepte, ou rexeite, a súa petición. Á súa vez, o usuario podo cancelar a petición en calquera momento premendo en ”Cancelar petición”.

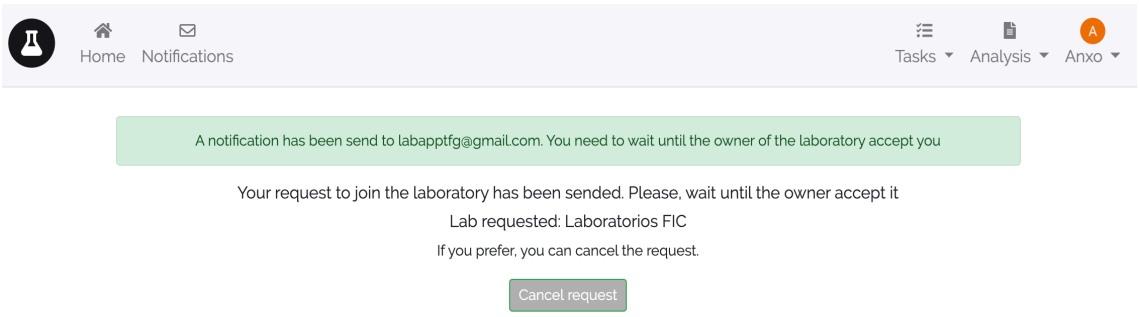


Figura B.13: Unirse a un laboratorio

B.4 Análises

B.4.1 Crear análises

Para proceder a creación dunha análise, é preciso despregar o menú de análises da barra superior. A opción indicada neste caso é ”Crear nova análise”.

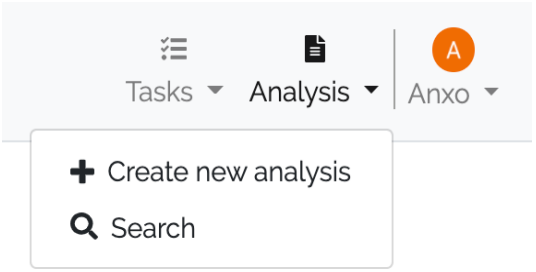


Figura B.14: Menú de análises

Unha vez seleccionada a opción, móstrase un formulario para a creación da análise cos diferentes campos posibles.

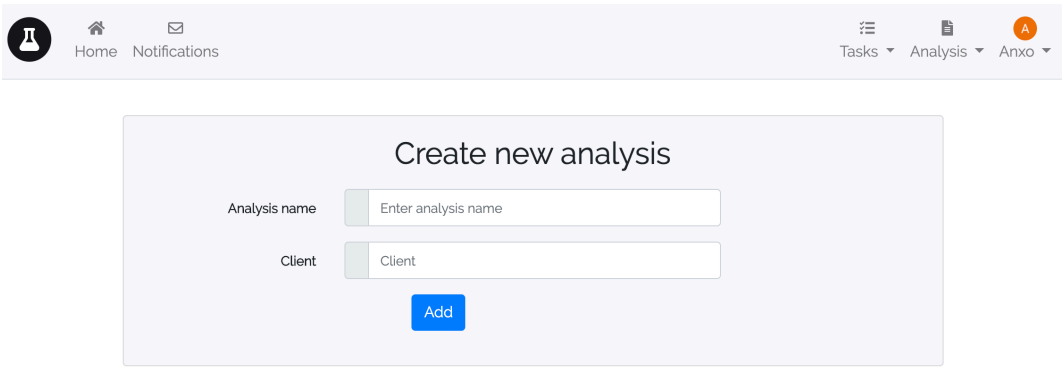


Figura B.15: Crear análise

Unha vez cubertos os campos e, tras premer ”Engadir”, o sistema mostra unha páxina de detalle da análise creada, coa información pertinente.

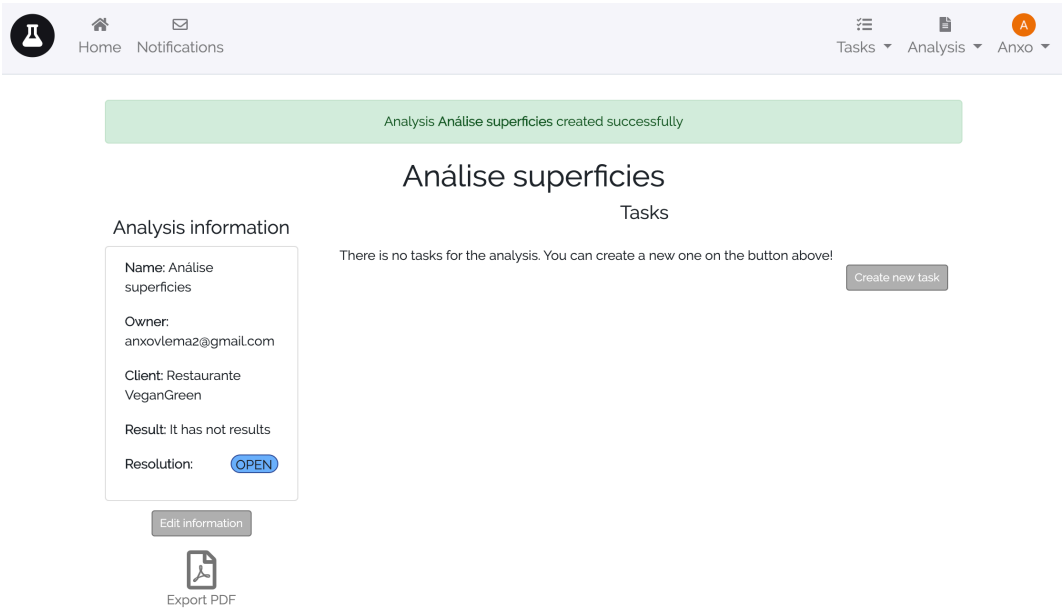


Figura B.16: Detalle dunha análise

Premendo no botón de ”Editar información” da análise, desprégase un menú coas diferentes opción a realizar sobre a análise.

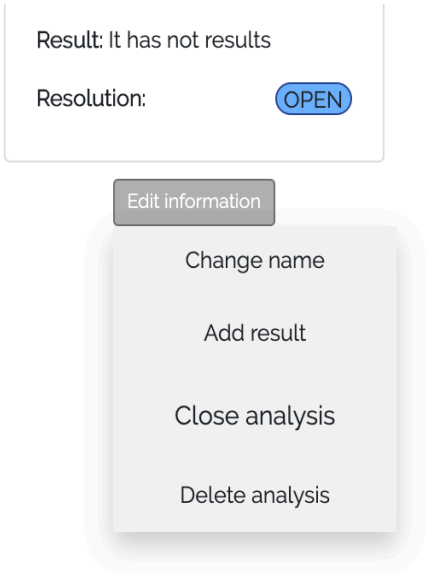


Figura B.17: Menú de opcións dunha análise

B.4.2 Opcións sobre unha análise

Cambiar nome

Dentro do menú de opcións, premendo ”Cambiar nome”, móstrase unha modal con un formulario para actualizar o nome da análise.

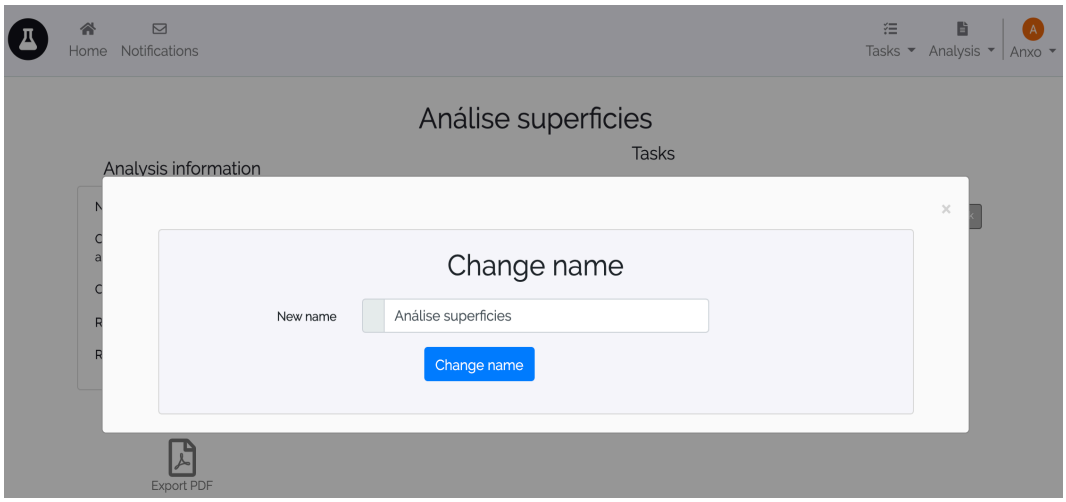


Figura B.18: Cambiar nome dunha análise

Engadir resultado

Mediante a opción do menú da análise, premendo ”Engadir resultado”, móstrase unha modal para realizar a dita acción.

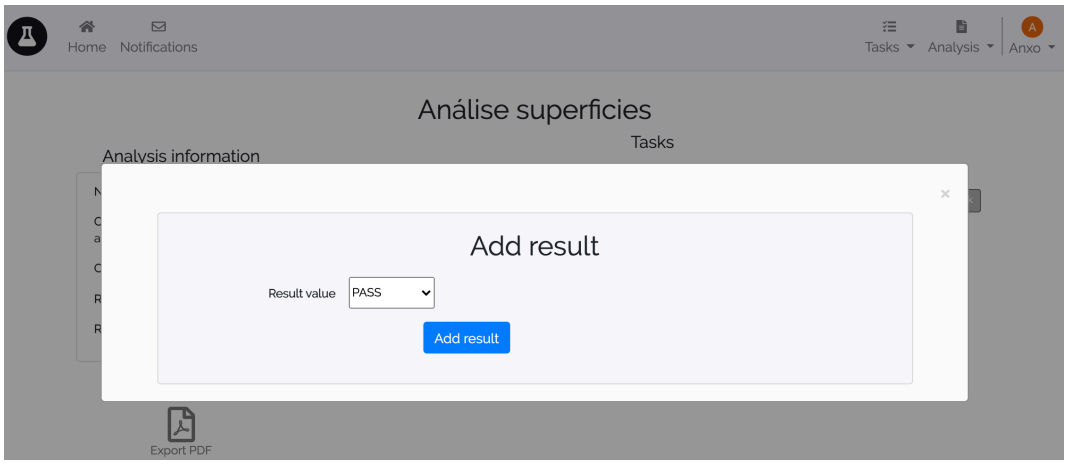


Figura B.19: Engadir resultado a unha análise

Pechar análise

Para pechar a análise, prémese no botón do menú de análise ”Pechar análise”. A análise péchase automaticamente, sempre que cumpra as condicións necesarias definidas no caso de uso.

Eliminar análise

Mediante a opción do menú da análise, premendo ”Eliminar análise”, móstrase unha modal para poder eliminar a análise, advertindo da acción que se vai realizar.

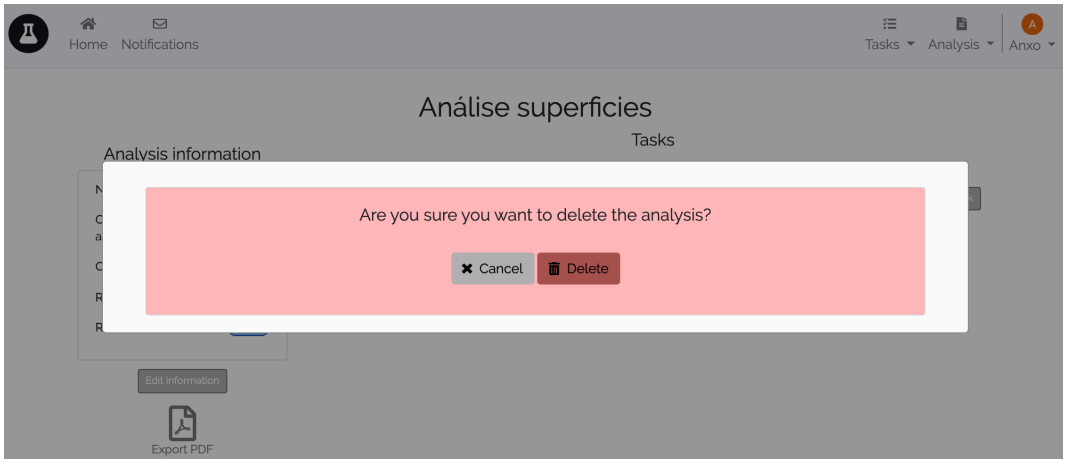


Figura B.20: Eliminar unha análise

B.4.3 Exportar análise a PDF

No detalle da análise, móstrase unha icona con texto ”Exportar PDF”. Premendo en dito botón o sistema descarga un PDF co resumo da análise.

B.5 Tarefas

B.5.1 Crear tarefas

Para proceder a crear unha tarefa é necesario premer o botón ”Crear tarefa” que se encontra no detalle dunha análise.

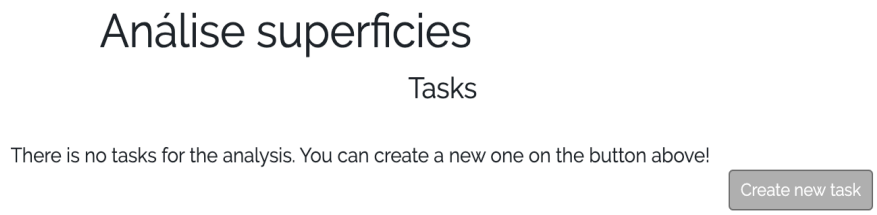


Figura B.21: Botón crear tarefa

A continuación, móstrase un formulario de creación da tarefa, cos diferentes campos necesarios, no cal se indica a análise para a cal vai ser creada a tarefa.

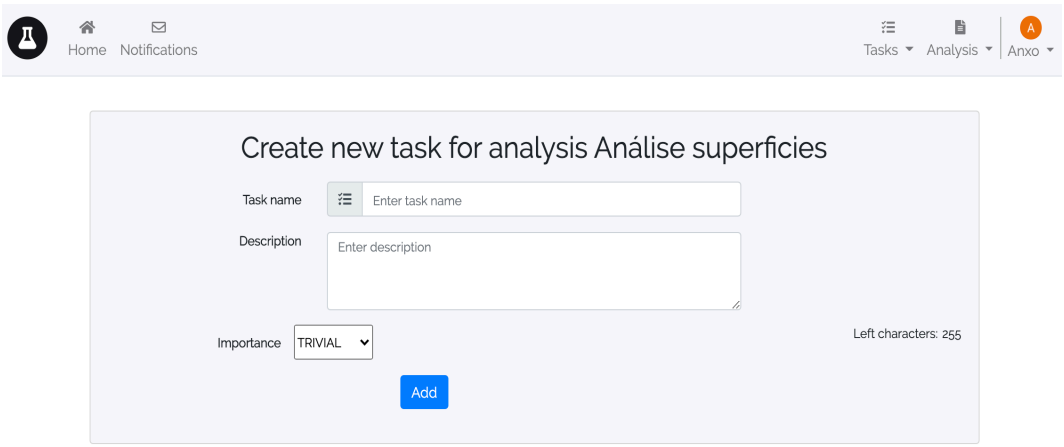


Figura B.22: Crear unha tarefa

Unha vez cubertos os campos necesarios e, tras premer no botón de ”Engadir”, móstrase de novo a páxina dos detalles da análise, coa nova tarefa na lista de tarefas.

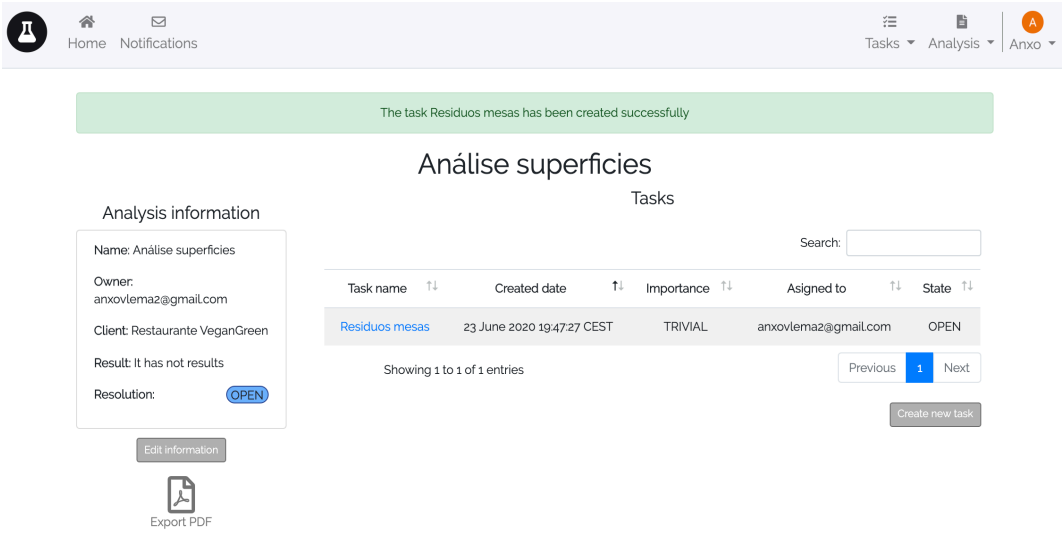


Figura B.23: Análise con tarefa

Para acceder ós detalles da tarefa, é preciso premer no nome da mesma.

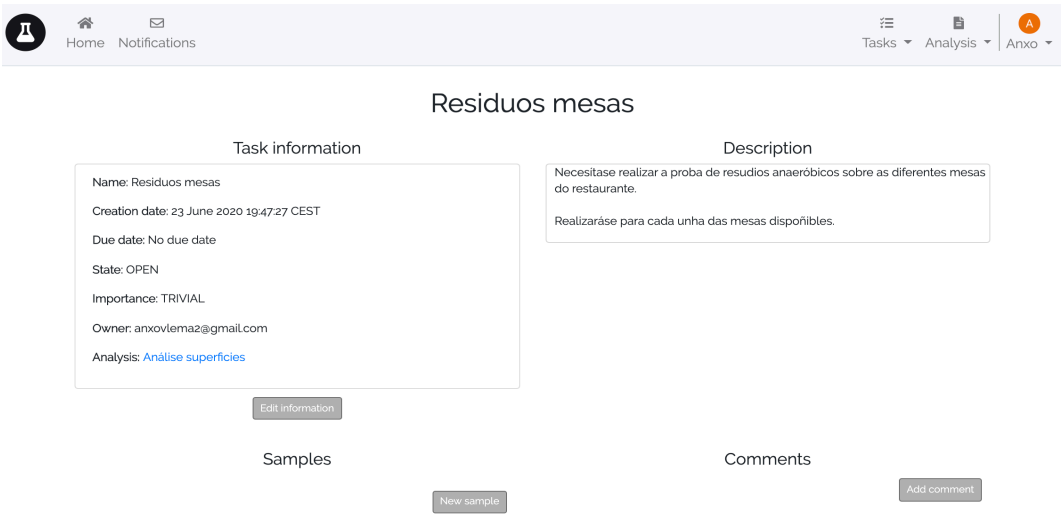


Figura B.24: Detalles dunha tarefa

B.5.2 Opcións sobre unha tarefa

Premendo no botón ”Editar tarefa” da páxina de detalles da tarefa, desprégase un menú coas diferentes accións a realizar sobre a mesma.

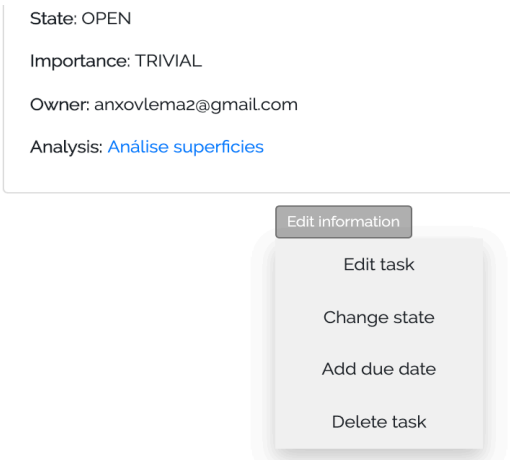


Figura B.25: Opcións sobre unha tarefa

Editar tarefa

Para editar os datos básicos dunha tarefa, o usuario preme no botón do menú de tarefas ”Editar tarefa”. Móstrase unha modal cos datos actuais coa posibilidade de modificalos.

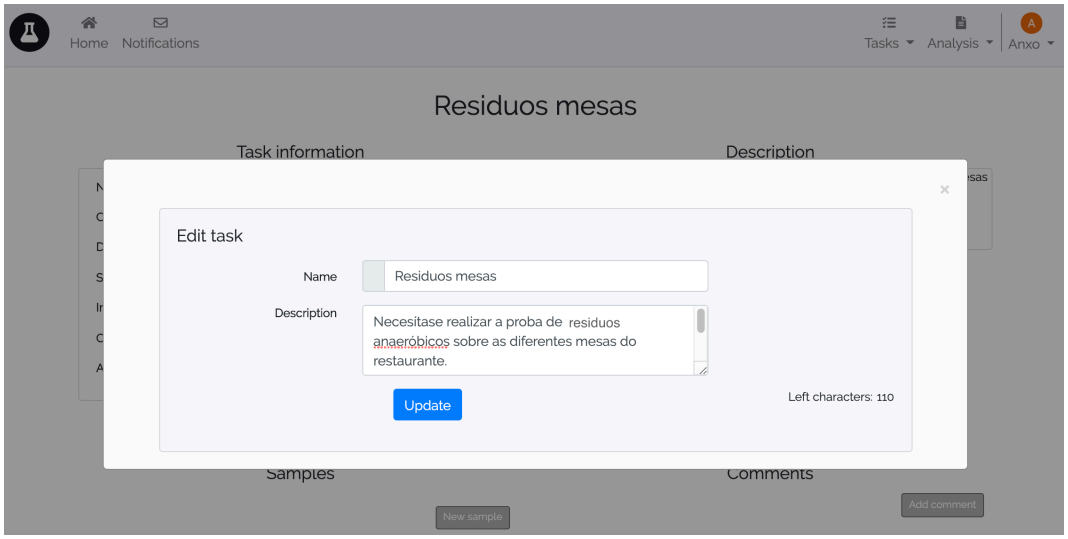


Figura B.26: Editar tarefa

Cambiar estado

Premendo no botón ”Cambiar estado” do menú de tarefas, móstrase unha modal na cal se permite modificar o estado da tarefa.

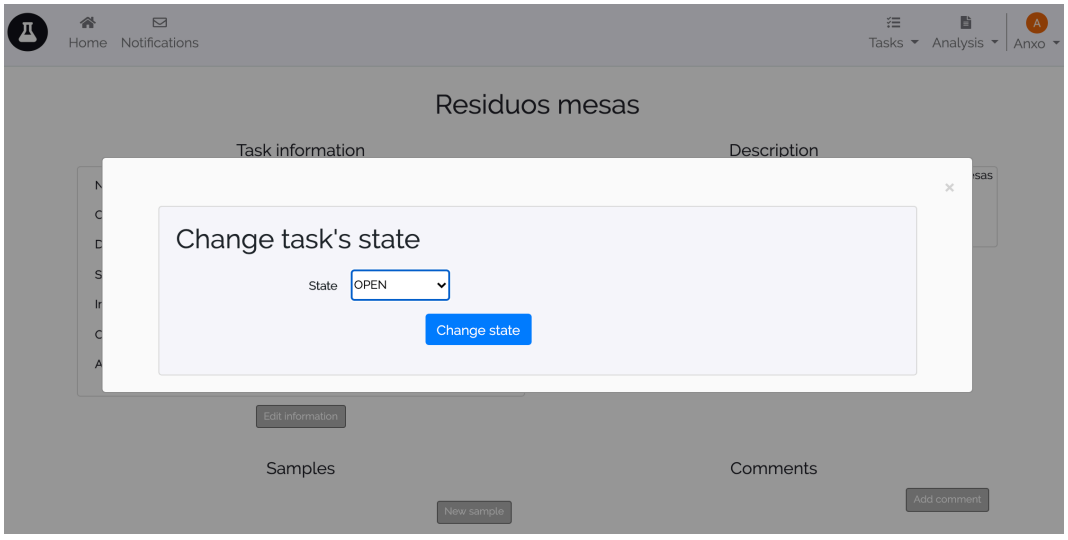


Figura B.27: Cambiar estado dunha tarefa

Engadir data de finalización

Para engadir unha data de finalización da tarefa, o usuario preme no botón ”Engadir data de finalización”, mediante o cal se mostra unha modal para seleccionar dita data. Unha vez engadida a data o usuario engádeselle no calendario de Google o evento.

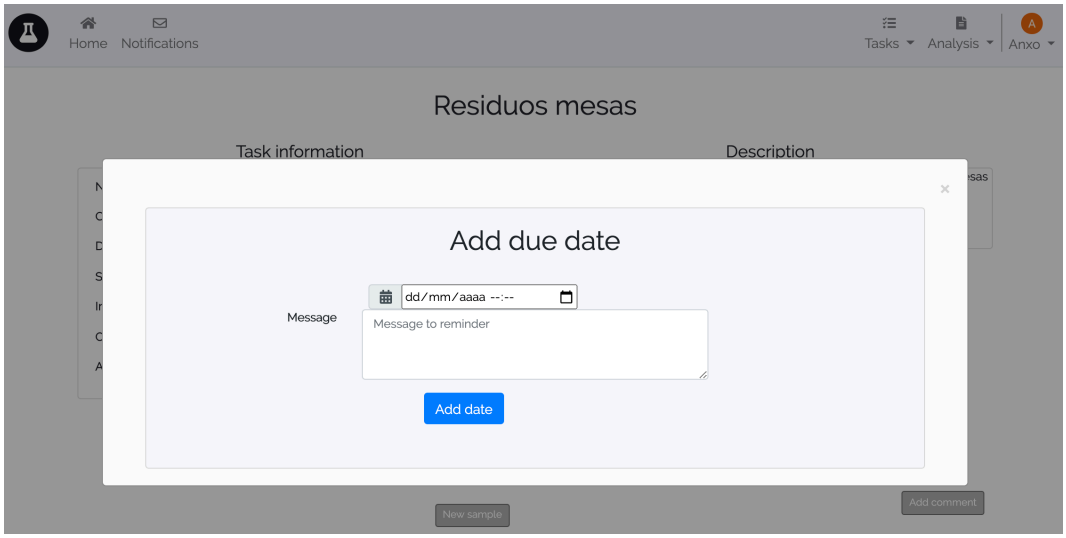


Figura B.28: Engadir data de finalización dunha tarefa

Eliminar tarefa

Por último dentro das opcións do menú de tarefas, para eliminar a tarefa é preciso premer no botón ”Eliminar tarefa”, co cal se mostrará unha modal solicitando a confirmación de dita acción.

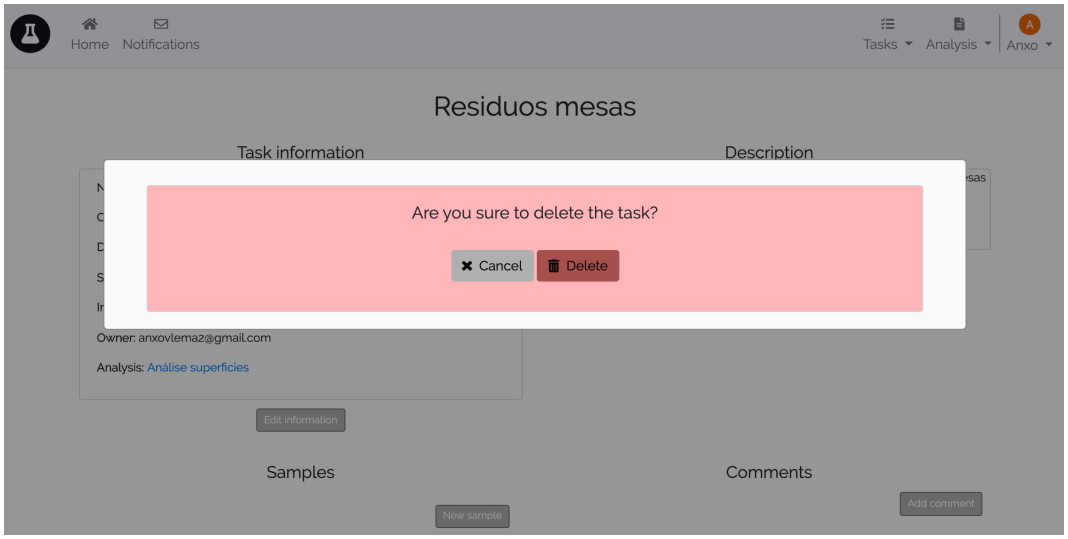


Figura B.29: Eliminar tarefa

B.5.3 Comentar nunha tarefa

Engadir comentario

Para comentar sobre unha tarefa, o usuario preme o botón ”Engadir comentario” situado na sección de comentarios no detalle da tarefa. Móstrase unha modal para engadir o comentario.

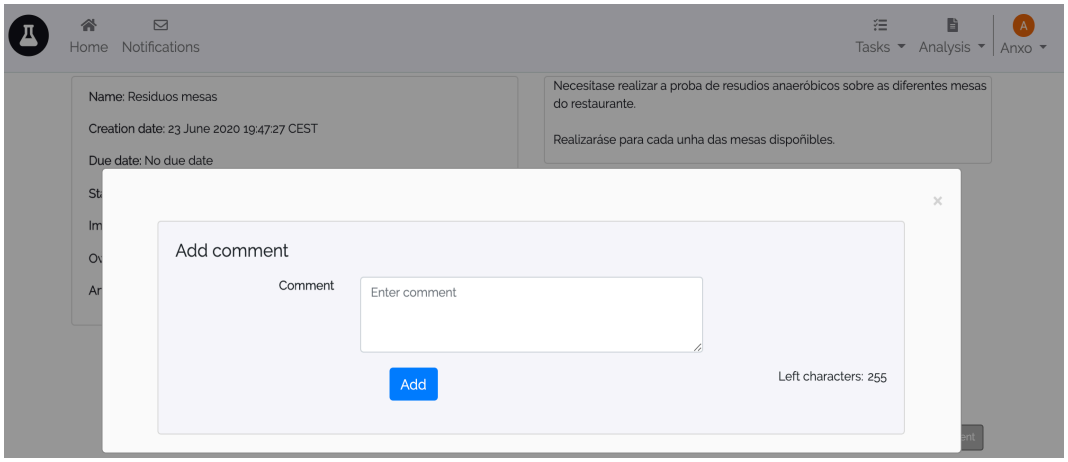


Figura B.30: Engadir comentario nunha tarefa

Unha vez engadido o comentario, mostrarase en forma de lista na sección de comentarios

da tarefa.

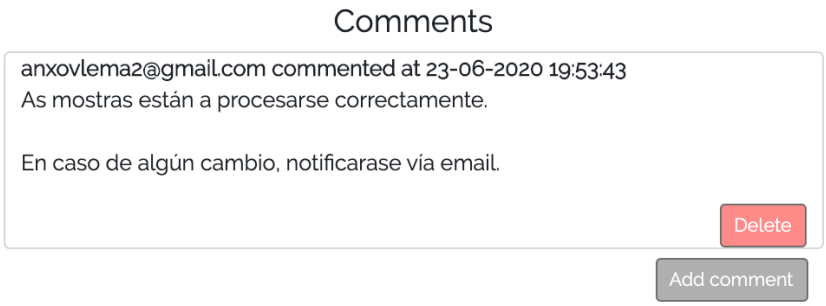


Figura B.31: Detalle dun comentario nunha tarefa

Eliminar comentario

Para eliminar dito comentario, prémese no botón ”Eliminar comentario”, co cal se mostra unha modal solicitando unha confirmación para a eliminación.

B.5.4 Ver as tarefas abertas

Para visualizar as tarefas abertas do usuario, na barra superior despégase o menú de tarefas.

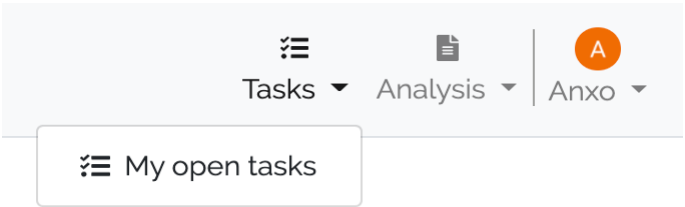


Figura B.32: menú de tarefas

A continuación, o usuario preme no botón ”Ver as miñas tarefas abertas”, co cal se mostra unha nova páxina cunha listaxe de tarefas en estado aberto, ou en progreso, que ten o usuario.

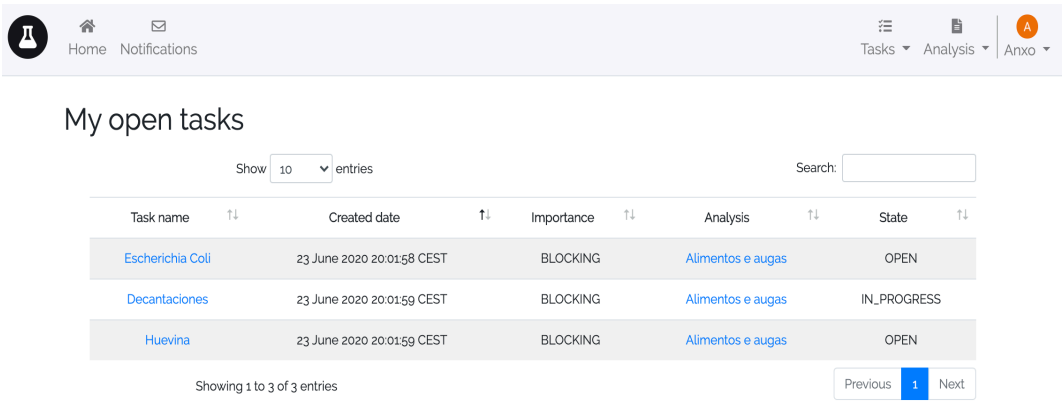


Figura B.33: Tarefas abertas

B.6 Mostras

B.6.1 Engadir mostra

Para engadir unha nova mostra a unha tarefa, o usuario preme o botón ”Engadir mostra”, mediante o cal se abre unha nova páxina con un formulario de creación de mostra.

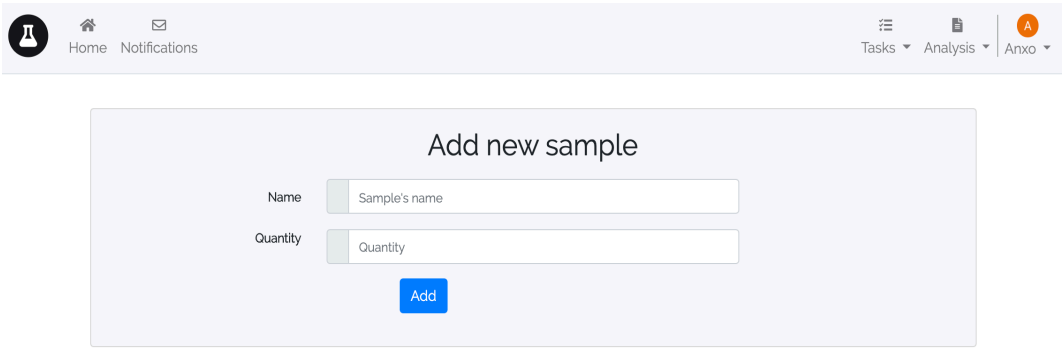


Figura B.34: Crear mostra

Unha vez engadida a mostra, móstrase na páxina de detalle da tarefa.

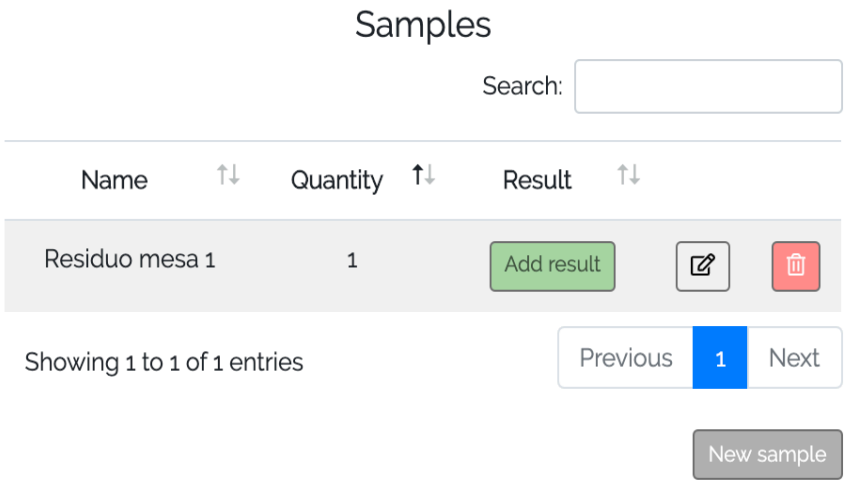


Figura B.35: Detalle da mostra

B.6.2 Engadir resultado a unha mostra

Para engadir un resultado a unha mostra, o usuario preme no botón ”Engadir resultado” da mostra que se desexe na lista de mostras da tarefa. Móstrase unha modal para inserir o resultado.

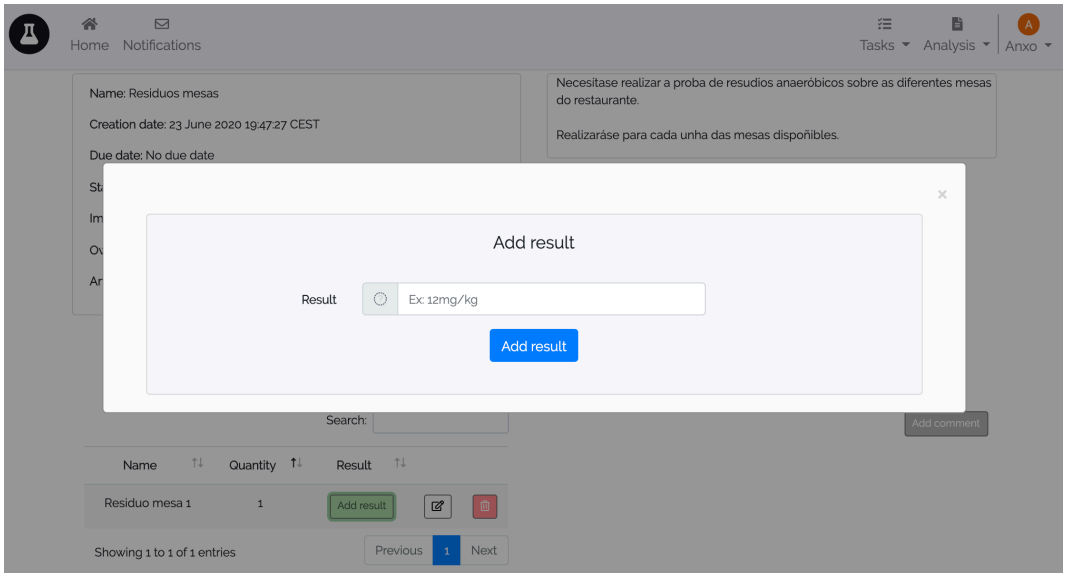


Figura B.36: Engadir resultado a unha mostra

B.6.3 Actualizar e eliminar mostra

Actualizar mostra

Para actualizar unha mostra, o usuario preme no botón con forma de papel e lapis na listaxe de mostras da tarefa. Móstrase unha modal cos campos a actualizar da mostra.

Eliminar mostra

Para eliminar unha mostra da listaxe de mostras da tarefa, é necesario premer o botón con forma de cubo de lixo. Móstrase unha modal solicitando confirmación para a eliminación.

B.6.4 Notificación

Para acceder as notificacións dun usuario é preciso premer na icona de notificacións da barra superior. Móstrase unha páxina cunha listaxe de notificacións.

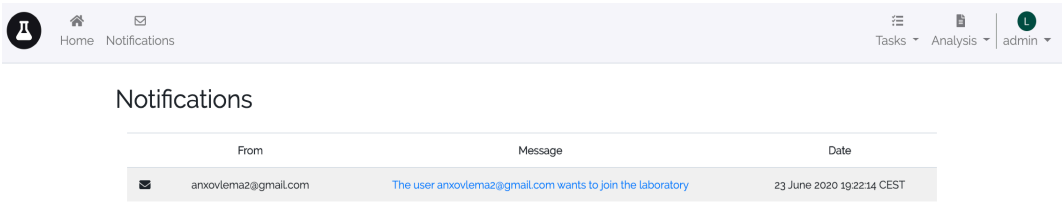


Figura B.37: Notificacións

B.7 Usuario LAB_MANAGER

A continuación móstranse as funcionalidades únicas dos usuarios ***LAB_MANAGER***.

Destacar que as funcionalidades anteriormente mostradas, son totalmente realizables por ambos usuarios, encargado e empregado.

B.7.1 Aceptar/rexeitar a asignación a un laboratorio

Estas solicitudes móstranse para o usuario que é propietario do laboratorio.

Accedendo as notificacións do usuario, pódese ver as distintas solicitudes para unirse ó laboratorio. Premendo nunha notificación móstrase o usuario que solicita unirse ó laboratorio. Premendo nun dos botóns de ”Aceptar” ou ”Rexeitar” a petición, acéptase ou rexéitase a solicitude.

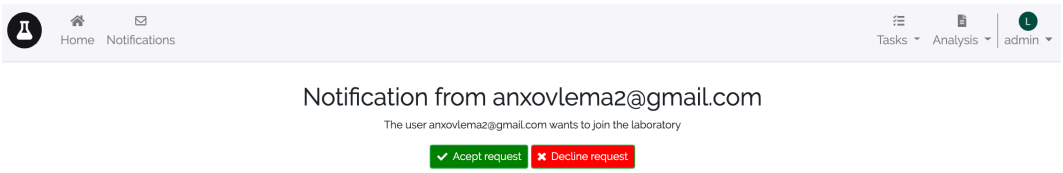


Figura B.38: Notificación de solicitude

B.7.2 Crear laboratorio

Os usuarios *manager* dispoñen da opción de crear un novo laboratorio. Para iso, o usuario accede, mediante o menú despregable do usuario, a opción ”Laboratorio”.

Móstranse dúas opcións, engadir novo laboratorio e buscar laboratorio. O usuario preme en ”Engadir novo laboratorio”.

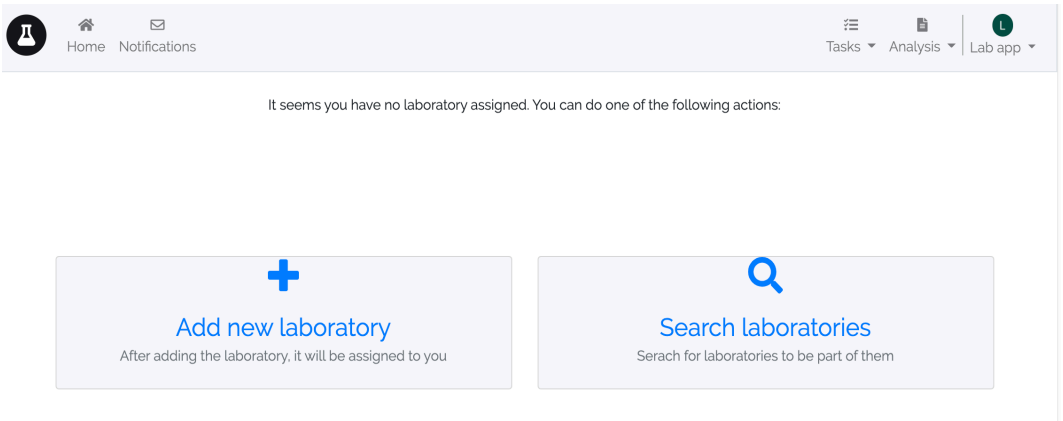


Figura B.39: Engadir laboratorio

A continuación, móstrase unha nova páxina con un formulario no que engadir o laboratorio.

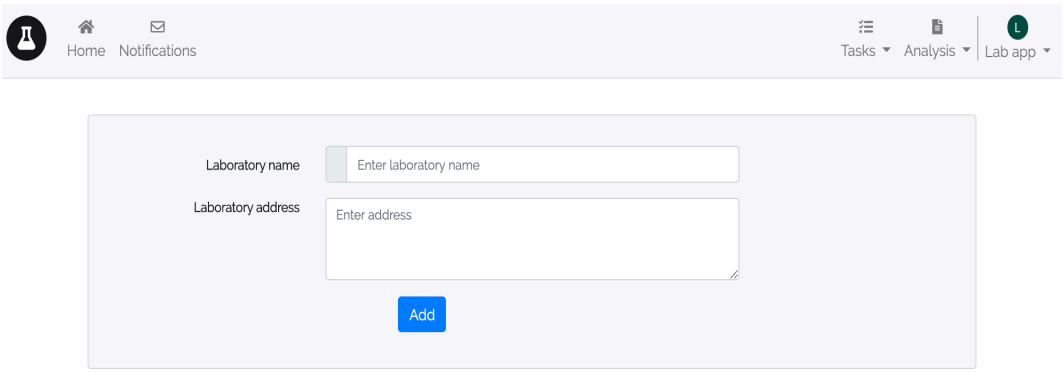


Figura B.40: Engadir laboratorio 2

B.7.3 Editar laboratorio

Os usuario *manager* dispoñen dun botón ”Editar laboratorio” na páxina de detalle do laboratorio.

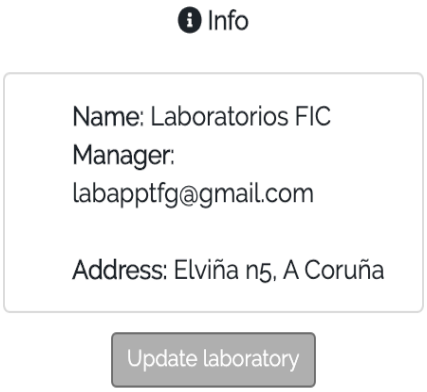


Figura B.41: Botón editar laboratorio

Mediante dito botón, móstrase unha nova páxina con un formulario para editar os datos do laboratorio.

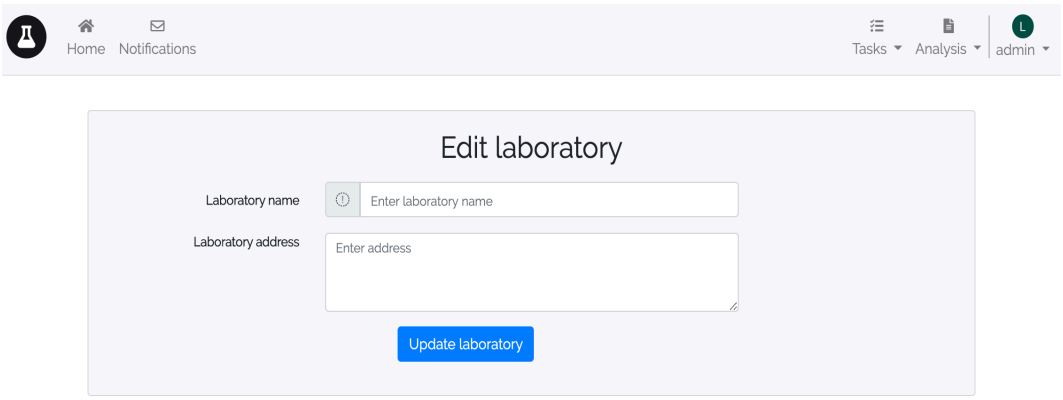


Figura B.42: Editar laboratorio

B.7.4 Eliminar perfil

En canto á eliminación do perfil dun usuario *manager*, existe un caso especial. Este caso dáse cando se elimina un usuario *manager* que é propietario dun laboratorio que ten máis usuarios que o propio propietario.

Para proceder a eliminación, accédese a información do perfil, como se explicou con anterioridade, e prémese o botón ”Eliminar perfil”.

Ao premer confirmar a eliminación, móstrase unha nova páxina, indicando que se necesita trasladar a propiedade do laboratorio a outro usuario.

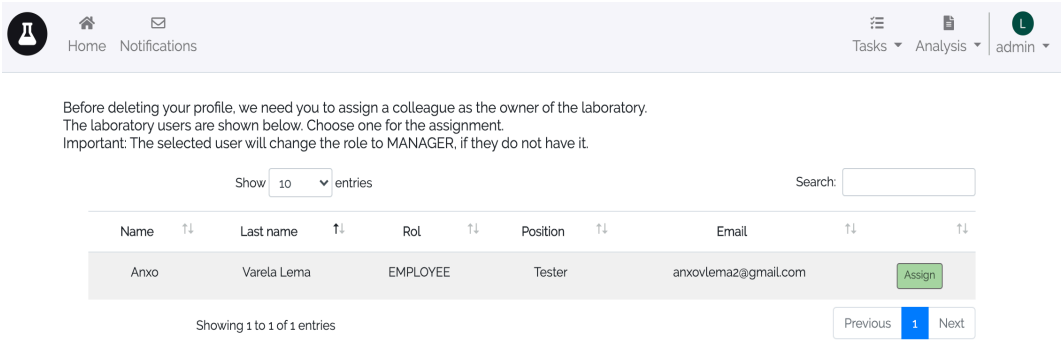


Figura B.43: Eliminar perfil manager 1

O usuario *manager*, debe seleccionar un novo usuario da listaxe para trasladarlle a propiedade do laboratorio. Unha vez prema o botón de ”Asignar”, móstrase unha modal de con-

firmación.

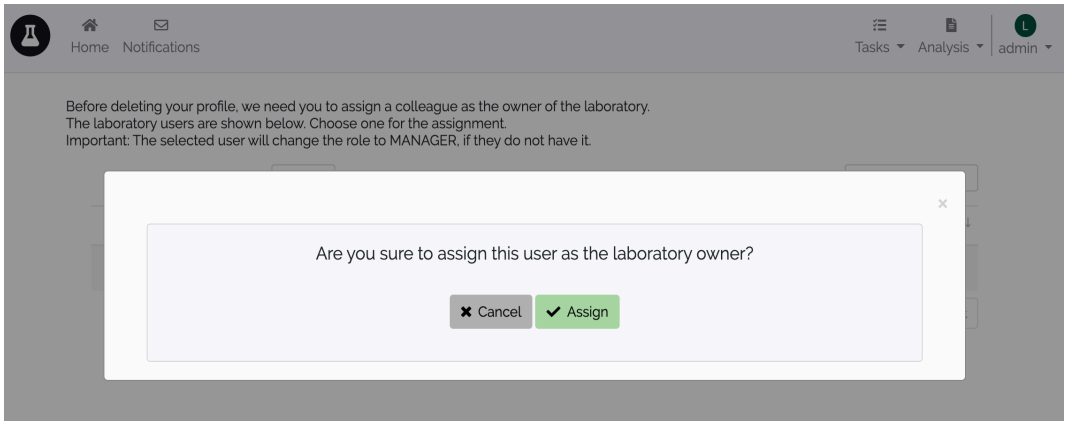


Figura B.44: Eliminar perfil manager 2

Tras confirmar a acción, elimínase o perfil e trasládalle a propiedade do laboratorio o novo usuario seleccionado.

Relación de Acrónimos

AOP *Aspect Oriented Programming.*

API *Application Programming Interface.*

CLI *Command-Line Interface.*

CRUD *Create, Read, Update, Delete.*

CSS *Cascading Style Sheets.*

DOM *Document Object Model.*

ECMA *European Computer Manufacturers Association.*

HTML *HyperText Markup Language.*

HTTP *Hypertext Transfer Protocol.*

IBM *International Business Machines Corporation.*

IDE *Integrated Development Environment.*

IoC *Inversion of Control.*

JDK *Java Development Kit.*

JPA *Java Persistence API.*

JSON *JavaScript Object Notation.*

LIMS *Laboratory Information Management System.*

MVC *Model-View-Controller.*

ORM *Object- Relational Mapping.*

PDF *Portable Document Format.*

POM *Project Object Module.*

REST *REpresentational State Transfer.*

RMI *Remote Method Invocation.*

RUP *Rational Unified Process.*

SPA *Single-Page Application.*

SQL *Structured Query Language.*

URL *Uniform Resource Locator.*

W3C *World WideWeb Consortium.*

XHTML *eXtensible HyperText Markup Language.*

XML *Xtensible Markup Language.*

Glosario

endpoint Serie de *URLs* dun servizo REST ás que pode acceder un usuario mediante un navegador web.

feedback Resposta dos implicados no proxecto sobre o produto ofrecido, con indicacións e opinións dos propios implicados.

stand-alone Programa de software que non necesita ser instalado na propia máquina na que vai ser executado, nin necesita acceso a unha conexión de internet para poder executarse.

Bibliografía

[1] G. S. G. B. James Gosling, Bill Joy and A. Buckley, "The java language specification java se 8 edition," 2015. [En liña]. Dispoñible en: https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf

[2] D. Flanagan, *JavaScript: The Definitive Guide*. O'Reilly Media, Inc., 2006.

[3] T. L. Steve Faulkner, Arron Eicholz and A. Danilo, "Html 5.1 2nd edition," 2017. [En liña]. Dispoñible en: https://www.w3.org/TR/html51/single-page.html

[4] B. Bos, "Cascading style sheets level 2 revision 2 (css 2.2) specification," 2016. [En liña]. Dispoñible en: https://www.w3.org/TR/CSS22/css2.pdf

[5] "An introduction to latex." [En liña]. Dispoñible en: https://www.latex-project.org/about/

[6] J. H. e. a. Rod Johnson, "Spring framework reference documentation," 2004. [En liña]. Dispoñible en: https://docs.spring.io/autorepo/docs/spring-framework/5.0.0.M1/spring-framework-reference/pdf/spring-framework-reference.pdf

[7] e. a. Phillip Webb, Dave Syer, "Spring boot reference documentation," 2012. [En liña]. Dispoñible en: https://docs.spring.io/spring-boot/docs/current/reference/pdf/spring-boot-reference.pdf

[8] "Tutorial: Using thymeleaf," 2018. [En liña]. Dispoñible en: https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.pdf

[9] "Eclipse platform technical overview," 2006. [En liña]. Dispoñible en: https://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.pdf

[10] R. Bharathan, *Apache Maven Cookbook*, 1st ed. Packt Publishing, 2015.

[11] S. Chacon and B. Straub, *Pro Git*, 2nd ed. Apress, 2014.

[12] "Páxina oficial de mysqlworkbench." [En liña]. Dispoñible en: https://www.mysql.com/products/workbench/

[13] "Páxina oficial de overleaf." [En liña]. Dispoñible en: https://www.overleaf.com/about

[14] "Mysql 8.0 reference manual." [En liña]. Dispoñible en: https://dev.mysql.com/doc/refman/8.0/en/introduction.html

[15] "Get started with the calendar api." [En liña]. Dispoñible en: https://developers.google.com/calendar/overview

[16] A. Parecki, *OAuth 2.0 Simplified*, 1st ed. Okta, 2017.

[17] T. B.-L. *et al*, "Hypertext transfer protocol – http/1.1," 1999. [En liña]. Dispoñible en: https://www.w3.org/Protocols/HTTP/1.1/rfc2616.pdf

[18] M. Masse, *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. O'Reilly Media, Inc., 2011.

[19] D. S. . M. C. . M. D. R. Boone, "A database and web application based on mvc architecture," 2006. [En liña]. Dispoñible en: https://ieeexplore.ieee.org/document/1604744/references#references

[20] R. F. Grove and E. Ozkan, "The mvc-web design pattern p.127-130." [En liña]. Dispoñible en: https://www.scitepress.org/Papers/2011/32969/32969.pdf

[21] P. Kroll and P. Kruchten, *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*, 1st ed. Addison-Wesley Professional, 2003.

[22] "The repository pattern - docs.microsoft.com." 2010. [En liña]. Dispoñible en: https://docs.microsoft.com/gl-es/previous-versions/msp-n-p/ff649690(v=pandp.10)

[23] "The facade design pattern - problem, solution, and applicability - w3sdesign.com." [En liña]. Dispoñible en: http://w3sdesign.com/?gr=s05&ugr=proble

[24] A. Dearle, *Software deployment, past, present and future," in Future of Software Engi- neering*, 2007.